A NEW COMPUTER-BASED SPEECH THERAPY TUTOR FOR VOWEL PRODUCTION

J. M. TURNBULL

A thesis submitted in partial fulfilment of the
requirements of the Council for National Academic Awards
for the degree of Doctor of Philosophy

September 1991

Department of Electronic and Electrical Engineering,
Dundee Institute of Technology in collaboration with the
Department of Medical Physics, Ninewells Hospital, Dundee

Declaration

I declare that while registered as a candidate for the degree for which this thesis is presented I have not been a candidate for any other award. I further declare that, except where stated, the work contained in this thesis is original and was performed by the author.

Signed

James Turnbull

Abstract

Our primary mode of communication is via speech. Therefore, any person who has difficulty in producing understandable speech, for whatever reason, is at a great disadvantage. It is the rôle of the speech therapist to help such people to improve their speech production wherever possible. The aim of this work was to develop a computer-based speech therapy tutor for vowel production. The Tutor would be able to analyse monosyllabic utterances in real-time, extract the vowel portion and match this to a pre-determined template, and display the result with no appreciable delay.

A fully-working prototype has been developed which employs general principles of aircraft tracking in a novel way, to track the coefficients of the quadratic factors of the all-pole linear-prediction model for speech production. It is shown how tracking these parameters can be used to extract extended vowels from a monosyllabic utterance. It is also shown how the algorithm which is used to determine the optimum frame-to-frame tracks can be used to perform template matching. The real plane on which the parameters are tracked, the rs-plane, suffers from non-linear scaling of frequency measures. This leads to poor spectral resolution of the perceptually-important low-frequency parameters. To overcome this problem, the rs-plane can be warped in order that distance measures taken between points on the plane are more meaningful perceptually.

The Tutor is based on a personal computer (PC). In order that real-time operation can be achieved, the processing power of the PC is enhanced by the addition of a digital signal processor (TMS32020) board and a transputer (T800) board.

The prototype Tutor was developed with help and advice from Dundee Speech Therapy Service, Tayside Health Board, who also conducted a short pilot study of the Tutor.

Acknowledgements

The author would like to thank Dr R.I. Damper, of the University of Southampton, and Mr A.T. Sapeluk, of Dundee Institute of Technology, who gave encouragement, displayed much enthusiasm, and provided assistance throughout the duration of the project. Thanks are also due to Mr R. Rimmer, of the Department of Medical Physics, Ninewells Hospital, Dundee, for his help with ensuring that the Tutor conformed to specific safety standards.

The author would also like to thank the members of Dundee Speech Therapy Service, Tayside Health Board, for their contribution in the definition of the design goals and the Tutor's user interface and for conducting the clinical trials of the Tutor.

Finally, the author acknowledges Dundee Institute of Technology, and in particular the Department of Electronic and Electrical Engineering, for providing the author with a Research Assistantship and funding the project.

Courses and Conferences Attended

To help the author understand something about the unique nature of the speech signal, a course on Phonetics and Phonology, which was part of a second arts degree course on Linguistics, was attended at St. Andrews University. This course lasted eight weeks starting mid-October 1987 and comprised three one-hour lectures per week. Its principal contents were the mechanisms of human speech production and categorisation of phonetic elements.

A short course on Research Methods was arranged by Dundee Institute of Technology. This course lasted six weeks starting mid-January 1988 and comprised one two-hour tutorial each week.

The following conferences were attended:

- European Conference on Speech Technology, Edinburgh, Scotland, 1–4 September 1987.
- IEEE International Conference on Acoustics, Speech & Signal Processing, Glasgow, Scotland, 23–26 May 1989.
- The Fourth ISAAC International Conference on Augmentative and Alternative Communication, Stockholm, Sweden, 12–16 August 1990.

Finally, a Research Seminar was presented to the Vision, Speech and Processing Research Group, Department of Electronics and Computer Science, University of Southampton, on February 8th 1989.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Overview

This chapter gives a general introduction to the research undertaken and describes how the Thesis is structured.

1.1 Aims of the Research

The human race has many forms of communication at its disposal. Communication can be verbal or non-verbal, aural or non-aural. Verbal communication is the means of transferring information using words. Aural communication is the means of transferring information using the speech production organs. Examples of non-aural/verbal communication are writing and sign language. Speech falls into the category of aural/verbal communication. An example of aural/non-verbal communication is laughing (by which information about one's emotions is conveyed). Raising one's eyebrows in surprise is a non-aural/non-verbal means of communication.

Although man has many means of communication at his disposal, our primary mode of communication is via the aural/verbal mode, i.e. speech. Therefore, any person who has difficulty in producing understandable speech, for any reason, is at a great disadvantage. It is the rôle of the speech therapist to help such people to improve their speech production wherever possible.

This Thesis describes the development of a new approach to tracking spectral parameters of the speech signal, and details how this was incorporated into a speech therapy tutor for vowel production (hereby referred to as the *Tutor*). The method uses an alternative representation of the poles of the Linear Prediction (LP) model for speech production. The Tutor was developed with help and advice from Tayside Health Board's Speech Therapy Department.

1.2 Structure of the Thesis

Each chapter has been written as a self-contained report on a particular aspect of the Tutor. Each comprises an overview page which summarises the content of the chapter, the main body of the chapter, and a list of any referenced papers and texts.

The appendices comprise useful derivations, which have been removed from the main text to increase readability. The *User Manual*, which describes the operation of the Tutor, has also been included as an appendix.

The main body of the Thesis starts with Chapter 2, which reports on the current

use of computers in speech therapy and explains the need for a computer-based vowel trainer.

Chapters 3, 4 and 5 describe a novel approach to tracking spectral parameters of the speech signal. Chapter 3 introduces the technique while relating it to other spectral-tracking techniques (in particular formant tracking). Chapter 4 describes a simulation of how the technique could be used to extract extended vowels from consonant-vowel and consonant-vowel-consonant utterances. This chapter also describes the simple template-matching technique which was used by the simulation to make assessments of vowel quality. Also included is a discussion of a perceptually-mapped template-matching technique. Chapter 5 describes work to estimate the optimal order of the LP model for use in the Tutor. This is introduced by considering the abundant psychophysical evidence that suggests that as few as two formants are required to identify a vowel.

Chapters 6, 7 and 8 describe how the developed algorithms were implemented on a multi-processor system to give a Tutor capable of real-time operation. Chapter 6 describes the hardware used while Chapters 7 and 8 describe the technical details of the implementation.

Chapter 9 describes the user interface and the visual feedback modes that the Tutor offers, while Chapter 10 reports on the results of testing the Tutor in a clinical application. Chapter 11 concludes by discussing possible developments for the future.

Chapter 2

Current Research

Overview

This chapter will cover background information on computer-based aids in speech therapy.

Speech technology has several applications in aiding the disabled community. Speech synthesis systems can assist a person who is unable to produce understandable speech. Speech recognition systems can be used by the physically-disabled who can produce good speech to control various items (for example, household appliances).

Speech recognition techniques can also be used to augment speech therapy training of the vocally-inarticulate. Groups who could benefit from such training include the hearing-impaired and the speech-impaired. The speech-impaired group in turn includes people with articulatory impairments such as cleft palate, loss of articulatory control such as dysarthia, post-operative articulatory dysfunction and those suffering from psychologically-based impairments such as lisps.

This chapter reviews current technology and research in the field of computer-based speech therapy aids.

2.1 Computers in Speech Therapy

Electronic devices for speech therapy applications have been in existence for a number of years. The early devices used specific hardware for certain functions and were therefore inflexible. Recent devices are computer-based and are quite flexible in the style of visual feedback.

In 1968, Risberg [1] described a series of visual feedback displays which could be used for speech correction. A crude amplitude spectrum display was reported which could be used to provide feedback during the production of Swedish long vowels. This tool used a set of band-pass filters to perform the frequency analysis of the speech sound and employed glow discharge tubes as the display mechanism. The same display was used for a fricative indicator.

Other displays included an *s*-indicator, which used a meter and a light to indicate the correct articulation of the /s/ sound, an intonation indicator which displayed the fundamental frequency of the speech sound on a cathode-ray-oscilloscope or an instrument meter, and a nasalisation indicator which also used an instrument meter. The intonation meter used a throat microphone and the nasalisation indicator used a vibration sensor which was placed on the side of the nose.

Risberg points out that the visual display can never be a good substitute for speech control through hearing. Consequently the speech therapy aid can only be used during short training periods.

Many speech therapy aids that were later developed displayed the same information but used a computer interfaced to the speech analysing hardware to output the results on a general visual display unit. For instance, in 1973 Nickerson and Stevens [2] proposed a speech training system based on a PDP-8E computer (manufactured by Digital Equipment Corporation). This device could display information on fundamental frequency, loudness, and frequency content of the speech signal. The first two parameters could immediately be shown as graphs against time and the third as a form of frequency/amplitude plot which changed as the user spoke. This latter display was described as 'a "busy" display that may present information to the viewer at a rate that exceeds his capacity to absorb it'. The display of the first two

parameters was also incorporated into a computer game by which the subject guided a ball through a hole in a wall into a basket. Also included was a simple smiling or sad face to show if success was achieved. Once the basic speech training aid had been developed, it was passed over to the teachers at a school for the deaf for a period of two years. The results of this field test is reported by Boothroyd *et al.* [3] and the subsequently-developed displays are described by Nickerson *et al.* [4].

As an alternative to these acoustic representations, some researchers have endeavored to display information that is related to the position of the articulators of the vocal tract during the production of the speech sound. For instance, in 1974 Crichton and Fallside [5] described a method of applying the linear prediction model of speech production to the training of persons with impaired hearing. They used this model to make an estimate of the cross-sectional area of the vocal tract as a function of distance along the vocal tract (the area function). This graph is displayed during speech production and is compared to a target graph of a particular speech sound. The rationale behind this display is that it *should* supply information to the subject that can be easily understood.

The device was further developed to show the time evolution of the area function in a manner similar to that of the spectrograph [6]. This new display was termed the *areagraph.* One possible failing of this type of display is that it could be too "busy" as was Nickerson's frequency display, i.e. there could be too much information for the subject to reasonably digest. It is interesting to note that the area function was chosen rather than the areagraph for later field test of a vowel trainer [7].

Recently, in an effort to reduce the cost and complexity of speech training aids, efforts have been made to base them on widely-available microprocessors. For example, in 1986 Boyd *et al.* [8] described a system based on the BBC microcomputer which displayed a two-dimensional frequency-amplitude plot, on which the amplitude of discrete frequency points was indicated with a vertical line, and a real-time spectrogram which shows frequency along the vertical axis, time along the horizontal axis, and amplitude information is represented by one of 8 colours. The frequency analysis was performed by a 16-channel switched-capacitor filter-bank.

Black and Gailey [9] used the same hardware configuration but displayed an estimate of the vocal-tract area function.

An alternative approach to the information-intensive feedback mechanisms described above is to display a single parameter about the utterance, such as "good" or "bad". Kewley-Port *et al.* [10] and Cottle *et al.* [11] both use proprietary speech recognition hardware for this purpose and incorporate the devices into a speech training aid.

A relatively recent (1st November 1988) commercial speech therapy aid is IBM's SpeechViewer. This tool comprises an IBM PS/2 equipped with the SpeechViewer adapter card incorporating a Texas Instruments TMS32010 digital signal processor [12]. The software drives the enhanced graphics of the PS/2 with great effect and incorporates the analysis tools into games in order to maintain the interest of the subject. Tools offered include pitch estimation, voicing, amplitude, and amplitude/frequency displays. A clinical trial of SpeechViewer has been reported by Öster [13]. The results were very positive and due mainly to SpeechViewer's meaningful and motivational visual feedback methods. Öster also agrees with Risberg's observation that computer-based speech-therapy aids can be a good complement to conventional speech-therapy practices but must be used with restriction and with moderation.

A recent review of computer-based speech training aids by Bernstein *et al.* [14] suggests that if these tools are to be effective, they must be used with a therapist working within a curriculum. This view is also shared by Boothroyd *et al* [3]. Bernstein *et al.* also suggest that the therapist should have an adequate understanding of acoustic phonetics.

To summarise, current speech-therapy tools are based on readily-available microcomputers to display the analysis parameters in an interesting manner. The speech analysis can be carried out by dedicated hardware incorporated into the computer, or the processing power of the computer can be augmented by adding an auxiliary processor. The main aim is to provide a common base on which many different tools can be implemented with varying user interfaces.

When developing a speech therapy tutor, it must be ensured that the display does

not present information in a complicated manner, nor faster than the rate at which the information can be absorbed. The effective application of a computer-based tool also requires an effective speech program and adequate teacher preparation.

2.2 A New Speech Therapy Tutor for Vowel Production

After many meetings and discussions with members of Tayside Health Board's Speech Therapy Department, it was decided that, locally, the most applicable area for a computer-based speech therapy aid was in the correction of vowel production. The training aid should cope with vowels produced within a monosyllabic context or as isolated vowels. For monosyllabic utterances the vowel part must be extracted hence feedback must be deferred until after that utterance has been completed. For isolated vowels the Tutor should be able to show a measure of how close a speech sound is to a target as the utterance is being produced. A display should also be incorporated which gives an indication as to the continuity of the produced sound.

The following chapter describes a system by which parameters of the speech signal can be tracked in order to determine the frame-to-frame deviations of the speech sound. Subsequent chapters describe how this is utilised to produce a speech therapy tutor which offers the above described features.

2.3 References

[1] A. Risberg (1968) "Visual aids for speech correction", *American Annals of the Deaf*, **113**, 178–194.

[2] R.S. Nickerson and K.N. Stevens (1973) "Teaching speech to the deaf: can a computer help?", *IEEE Transactions on Audio and Electroacoustics*, **AU-21**, 445–455.

[3] A. Boothroyd, P. Archambault, R.E. Adams, and R.D. Storm (1975) "Use of

a computer-based system of speech training aid for deaf persons", *The Volta Review*, **77**, 178–193.

[4] R.S. Nickerson, D.N. Kalikow, and K.N. Stevens (1976) "Computer-aided speech training for the deaf", *Journal of Speech and Hearing Disorders*, **41**, 120–132.

[5] R.G. Crichton and F. Fallside (1974) "Linear prediction model of speech production with applications to deaf speech training", *Proceedings of the Institution of Electrical Engineers, Control & Science*, **121**, 865–873.

[6] F. Fallside and S. Brooks (1977) "Some displays for computer analysed speech", *Proceedings of the Institution of Electrical Engineers, Control & Science*, **124**, 1227–1229.

[7] S. Brooks, F. Fallside, E. Gulian, and P. Hinds (1981) "Teaching vowel articulation with the computer vowel trainer", *British Journal of Audiology*, **15**, 151–163.

[8] I. Boyd, W. Millar, L.M. Boyd, and E.L. Gailey (1986) "A spectral visual feedback system for the hearing impaired", *IEE International Conference on Speech Input/Output; Techniques and Applications, London*, Conference Publication No. 258, 257–262.

[9] N.D. Black and E.I. Gailey (1986) "Visual Feedback for the deaf", *Proceedings of the Institute of Acoustics*, **8**, 217–223.

[10] D. Kewley-Port, C.S. Watson, D. Maki, and D. Reed (1987) "Speaker-dependent speech recognition as the basis for a speech training aid", *Proceedings International Conference on Acoustics, Speech, and Signal Processing, ICASSP '87*, 372–375.

[11] M. Cottle, C. Gaunt, and R.M. Stephens (1989) "The use of speech recognition as an aid for speech therapists", *Proceedings Eurospeech '89, Paris*, 710–713.

[12] H. Crepy (1989) "SpeechViewer programming interfaces", *Document Number F138*, IBM France Scientific Center, Paris.

[13] A.-M. Öster (1989) "Applications and experiences of computer based speech training", *Proceedings Eurospeech '89, Paris*, 714–717.

[14] L.E. Bernstein, M.H. Goldstein, and J.J. Mahshie (1988) "Speech training aids for hearing-impaired individuals. I. Overviews and aims", *Journal of Rehabilitation Research and Development*, **25**, 53–62.

Chapter 3

Tracking Spectral Parameters of Speech

Overview

This chapter discusses methods for tracking spectral parameters of the speech signal, namely formant-tracking techniques. Tracking the poles on the z-plane is then examined as an alternative. It is shown, however, that pole-tracking has some not-unsurmountable problems which tend to complicate the tracking algorithm.

A method of obtaining the poles is then described, which introduces a two-dimensional real plane, the rs-plane. The properties of the rs-plane are explored and utilised to address the problems associated with tracking pole movements on the z-plane.

3.1 Introduction

The Tutor analyses speech sounds as they are produced and produces a set of parameters which can be related to frequency information. These parameters are tracked in order to determine their frame-to-frame deviation and to determine when a steady-state region of the utterance occurs.

The automatic tracking of objects involves associating the object to a previous existence of that object in order to update its temporal evolution. In the case of speech analysis, the objects that are tracked are the parameters of some model that represents the speech sound. The most commonly tracked parameters, representing a model for voiced sounds, are the fundamental frequency and the formant frequencies.

Tracking the fundamental frequency is a trivial problem once it has been determined: since there is only one parameter being tracked the association is a simple one-to-one correlation. However, the tracking of the formant-frequency movements throughout a section of voiced speech can be a more difficult task. The formant frequencies can be used to classify vowels (see Chapter 5) and are tracked to eliminate extraneous formants caused by noise or as a method of determining whether one apparent formant could actually be two closely-spaced formants.

The simplest formant tracking algorithms [1,2] determine a smoothed spectrum of the speech waveform, detect the first N peaks in the spectrum and assign the frequencies of the peaks to formants F_1 to F_N in a one-to-one fashion. An alternative to this is to segment the spectrum, either using non-overlapping [1] or overlapping [3] segments, using prior knowledge of likely positions of the formants, and perform peak picking within these segments. A more sophisticated peak picking algorithm has been proposed [4] which detects the first three formants. It uses a global optimisation scheme on the whole spectrum to eliminate the effects of local maxima.

More elaborate tracking strategies have been proposed [5,6] which detect vowel centres by identifying maxima in the signal energy where voicing occurs. These points in the speech waveform are used as *anchor points* from which a fixed number of formants are determined and subsequently tracked both forward and backward in time using continuity as the basis for discarding erroneous or spurious data.

An alternative strategy [7] also tracks the formants both backwards and forwards in time, but generates an arbitrary number of *strings* of continuous formant data. Spurious data is then eliminated by discarding strings which are shorter than a predetermined length.

Other formant-tracking techniques determine the formant frequencies directly from the speech waveform by estimating the parameters of a speech production model whose parameters are the formant frequencies. Examples of such methods are the use of the extended Kalman filter [8] and an *analysis-by-synthesis* technique [9]. Both of these techniques use models which have a fixed number of formants.

The above algorithms which track a fixed number of formants suffer from the necessity to discard possible formants: the speech signal spectrum does not have a constant number of peaks and hence the number of formants within the speech signal are not known prior to analysing the speech waveform. So a number is chosen, typically 3, above which additional formants are ignored. Also the algorithms which track the formants both backwards and forwards in time are basically unsuitable for implementing on a real-time system.

An alternative to tracking the formant frequencies would be to track the poles of the LP model for speech production. This would be desirable as the number of objects tracked is fixed and because the number of pole-pairs that represent formants can vary.

3.2 Tracking the Poles of the LP Model

The poles of the LP model can be related directly to the positions of the formant frequencies of the speech sound in the frequency domain [10]. Therefore, the positions and trajectories of the poles on the z-plane could supply adequate information to enable the voiced speech sound to be identified. Rather than converting the information about the poles on the z-plane into formant frequency information and *then* tracking the formants, it should be possible to track the pole movements directly.

This would involve tracking objects on what is physically a two-dimensional space.

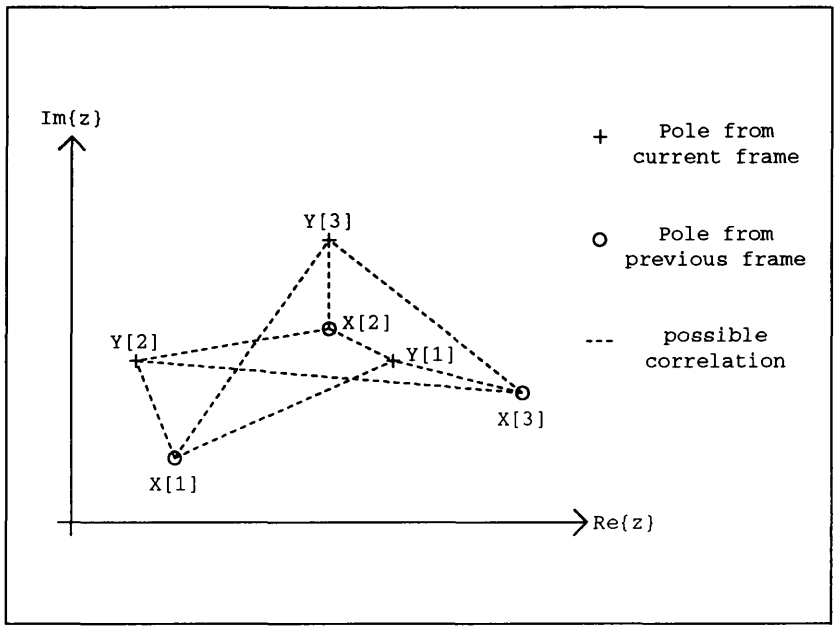

Figure 1: Hypothetical frame-to-frame correlation problem.

The tracking procedure described in this section was originally intended for the tracking of aircraft [11] but is equally applicable to the present problem.

Tracking requires that each pole of the current frame be related to all poles of the previous frame. Simple association of poles which are closest between consecutive frames can result in erroneous matching as illustrated in Figure 1, which shows three z-plane poles from two successive frames, X and Y.

If a match is made solely on the distance between X and Y poles, then the end result will depend on the order in which the poles are taken. Taking pole $Y[1]$ first, $X[2]$ is the nearest and would therefore be taken to correlate with pole $Y[1]$. Considering pole $Y[2]$ next, $X[1]$ is taken as a match. However, when it comes to $Y[3]$, it too should correlate with $X[2]$ on the basis of minimum distance and pole $X[3]$ would be left without a match. In fact, $Y[1]$ and $X[3]$ should have been associated as a matching pair in the first place.

Such errors can be completely avoided by calculating a distance for every possible permutation of correlations, and then selecting the set which associates each pole in X with each pole in Y in one-to-one fashion so as to minimise the cumulative distance.

Permutation	Correlate for $Y[1]$	Correlate for $Y[2]$	Correlate for $Y[3]$
1	$X[1]$	$X[2]$	$X[3]$
2	$X[1]$	$X[3]$	$X[2]$
3	$X[2]$	$X[1]$	$X[3]$
4	$X[2]$	$X[3]$	$X[1]$
5	$X[3]$	$X[1]$	$X[2]$
6	$X[3]$	$X[2]$	$X[1]$

Table 1: Possible permutations of correlations.

Consider again the hypothetical situation shown in Figure 1. Each pole in Y has three possible correlates, and so there are $3 \times 3 = 9$ distances to be considered. Table 1 shows all the permutations of possible correlations. For the permutation on each row, the correlates for $Y[1]$, $Y[2]$, and $Y[3]$ are shown.

For the three pole system there are $3! = 6$ permutations. For each permutation, the sum of the three distances corresponding to the three pairs of correlates is calculated to give the cumulative distance. The optimal permutation is the one with the smallest total distance metric. The tracking algorithm can be summarised thus:

- generate a table of all distance metrics for each Y to every other X ;
- for each permutation calculate the cumulative distance;
- select the permutation with the smallest cumulative distance.

This may be feasible for model orders up to about 5, where the number of permutations considered is $5! = 120$, but for a larger model order of, say, 10 the number of permutations would be $10! = 3628800$. This algorithm is therefore ideal for small model orders, since the optimum correlation is guaranteed, but for larger orders, which are common in speech analysis applications, such an algorithm is inappropriate.

One method of reducing the number of permutations which have to be considered is to disregard associations with poles that are further than a set distance away, and subsequently the best correlation is chosen [11]. This approach, however, is complicated and cannot guarantee that the optimum correlation is chosen.

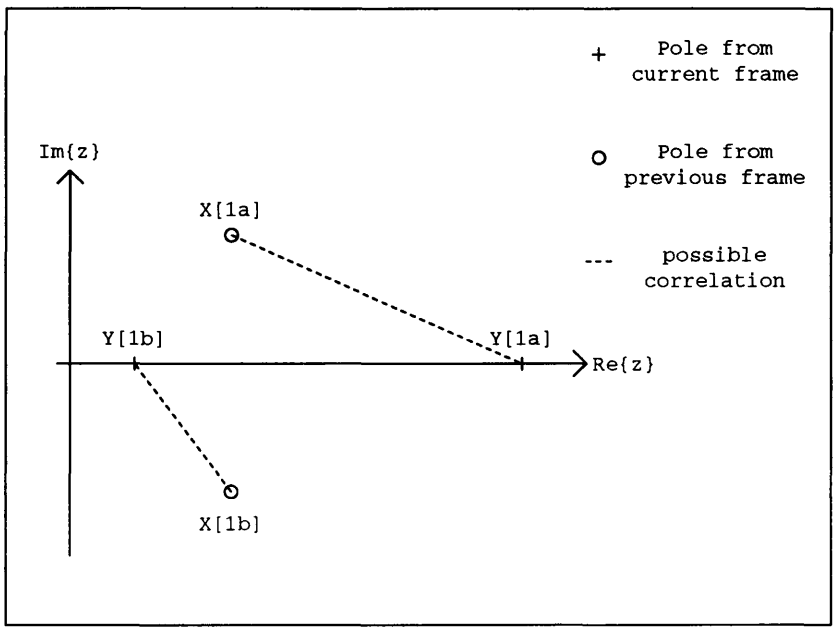

Figure 2: Unpredictable correlation when complex poles become real.

Another method of reducing the computation would be to consider the axial symmetry of the z-plane. If all the poles occurred in complex conjugate pairs then only half the number of poles would need to be considered. The number of permutations would be reduced from $p!$ to $(p/2)!$, which is a very worthwhile reduction. However, the poles do not always occur in complex-conjugate pairs: depending on the model order, it is common for one or more pairs of roots to be real. Therefore if the tracking is restricted to only one half of the z-plane, there need not always be the same number of poles to be correlated between consecutive frames. One or more poles would be left without a match which would cause the tracking process to fail.

An additional consequence of complex pole-pairs which move onto the real axis is illustrated in Figure 2. Pole pair $X[1a]$ and its complex conjugate $X[1b]$ move onto the real axis in consecutive frames, resulting in the real pole-pair $Y[1a]$ and $Y[1b]$. It is difficult to predict which of the two combinations, $X[1a]$ and $X[1b]$ matching with $Y[1a]$ and $Y[1b]$ respectively, or $X[1a]$ and $X[1b]$ matching with $Y[1b]$ and $Y[1a]$ respectively, will be chosen. This would complicate any recognition process which attempted to match tracks from one utterance with those of another.

In summary, the problem of tracking the poles of the linear prediction model in

the z-plane has been examined. Although difficulties have been identified, they are not unsurmountable. However, they do tend to complicate the tracking algorithm.

3.3 Obtaining the Poles of the LP Model

In order to track the poles of the LP model, the roots of the denominator polynomial must be obtained. Equation 1 gives the transfer function (TF) for the LP model for speech production.

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{1}$$

To obtain the poles in terms of z rather than z^{-1} the numerator and denominator are multiplied by z^p to give Equation 2.

$$H(z) = \frac{Gz^p}{z^p - \sum_{k=1}^{p} a_k z^{p-k}} \tag{2}$$

The numerator of Equation 2 has only a z^p term, which gives p zeros all situated at the origin. These are trivial zeros, and for this reason the LP model is considered to be an all-pole model. The poles of the TF are the roots of the denominator polynomial. If the order of the polynomial is greater than 4, there is no analytical method for finding its roots and an iterative method must be used. The coefficients of the polynomial are generated from the LP analysis of the speech waveform. As with any realisable system, the coefficients are real. Hence, the roots of the polynomial occur in pairs of real numbers or complex conjugates.

Algorithms which find one root at a time, such as the Newton-Raphson method, must be implemented using complex arithmetic. One method which is suited to this application is Bairstow's method.

3.3.1 Bairstow's Method

Bairstow's method [12] finds quadratic factors of a real polynomial, which in this case is in terms of z. The quadratic factors are of the form $(z^2 - rz - s)$ such that

the division of the polynomial by this quadratic gives a remainder of zero. As the coefficients of both the quadratic factor and the original polynomial are real, only real arithmetic is required.

Once one quadratic factor of the polynomial has been obtained, the polynomial is deflated by dividing it by the factor. Bairstow's method is then applied to the resulting polynomial and this process is repeated until the order of the remaining polynomial is less than or equal to 2. The roots are then found analytically from the quadratic factors using the quadratic formula.

As with all iterative methods an initial estimate is required. This method, however, suffers from divergence if a reasonable initial estimate of the coefficients of the quadratic factor is not given [12,13]. The remainder of this section addresses the problem of obtaining reasonable initial estimates and considers how divergence problems can be minimised.

3.3.2 Obtaining Initial Estimates for Bairstow's Method

For slowly-varying sounds such as the vowels, semivowels, and glides, it is a reasonable assumption that the poles of the LP model will lie within the unit circle on the z-plane. Thus an initial estimate should be taken from somewhere on or within the unit circle and these z-plane parameters mapped to initial estimates of the coefficients of the quadratic factor, r and s.

The iterative process is terminated when the change in the coefficients r and s for consecutive iterations become negligible or when the number of iterations exceeds some pre-determined maximum. If the process is terminated by the latter constraint, then the method has been unsuccessful on that attempt and the roots are deemed unreliable; another initial estimate of r and s must be made.

To investigate the suitability of Bairstow's method for this application, this algorithm was coded in Pascal. A large data set was constructed by analysing vowel sounds using a 10th-order LP model (the LP analysis method employed is mentioned in Section 4.2). The root finding routine was verified by reconstructing the polynomials from the quadratic factors obtained and comparing these with the original data set.

For this method the poles of the LP filter were found successfully about 70% of the time. However, for the majority of the times that the algorithm failed, the termination of the process was not due to exceeding a maximum number of iterations but to a floating-point overflow. This was caused by the iterative method being rapidly divergent on these occasions.

One method of making the algorithm more robust would be to monitor the magnitude of r and s in order to detect divergence. As was stated previously, the poles of the LP model should not lie outside the unit circle. The direct application of this as a test for divergence would require that the poles be calculated during the extraction of the quadratic. To avoid root finding at each stage, the interior of the stability circle can be mapped onto a two-dimensional real space, the dimensions of which are the coefficients of the quadratic factor [14,15], r and s.

This mapping is not widely used, and is generally only mentioned as a stepping stone to the stability circle on the z-plane. However, in this application, it is ideal as the stability of a second-order system can be determined directly from the r and s coefficients [16].

It should be noted that this representation was independently developed by the author until it was discovered in the literature. For this reason, the author's representation differs slightly from that in the literature in that it is rotated by 180 degrees.

The region of stability on this coefficient plane, henceforth referred to as the rs-plane, is shown in Figure 3. After each iteration in the root-finding algorithm, if the point $[r, s]$ lies some distance outside this triangle, then it is a reasonable assumption that the process is becoming divergent.

3.3.3 Limiting Divergence

It is essential, both for computational efficiency and for robustness, that any divergent tendency during iteration is detected and stopped. Various methods of curbing this divergence were tried.

The first method tried was to limit r and s such that if, after an iteration, the current estimates of r and s lie outside the stability triangle then they were brought

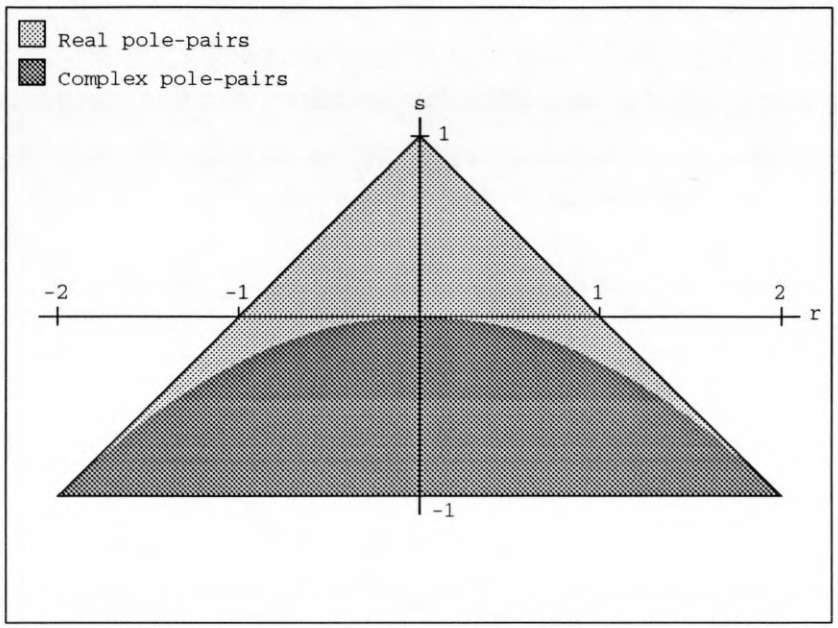

Figure 3: Region of stability on the rs-plane.

back into the triangle. This can be achieved by, first, limiting s to the range $-1 \leq s \leq 1$. This clips the point either onto the horizontal edge of the triangle or nearer to one of the sloping edges. The estimate can then be tested for lying outside the triangle and further clipped if necessary. If $(s > (1 - r))$, which corresponds to the estimate lying outside the right sloping edge of the triangle, then r is set equal to $(1 - s)$, which brings the estimate back to this edge. Similarly, if $(s > (r + 1))$ then r is set equal to $(s - 1)$.

This method was successful in eliminating the overflow conditions. However, a large percentage of the time the estimate would get *trapped* in the top corner of the triangle. This happens when taking an estimate from this corner produces an estimate which is subsequently clipped back to the corner. This process will continue until the maximum number of iterations is reached, which is an unsatisfactory situation.

The object of the clipping is to prevent any divergence from running long enough to give a computational overflow. It is therefore not necessary always to restrict the estimate to within the stability triangle. The simplest continuous boundary which contains the stability triangle is a circle of radius $\sqrt{5}$ centered about the origin.

Using this restriction, if the distance from the center of the circle to the current

Test Criterion	Implementation
outside triangle	$(s < -1)$ OR $(s > (1 + r))$ OR $(s > (1 - r))$
Euclidean distance	$(s^2 + r^2) > 5$
Chebychev distance	$(\|r\| + \|s\|) > 3$

Table 2: Test criteria for detecting divergence.

estimate, $d = \sqrt{r^2 + s^2}$, is greater than $\sqrt{5}$ then both r and s are scaled down by a factor of $d/\sqrt{5}$.

This second method also eliminated the overflow conditions. Again, however, a satisfactory result would very often not be achieved before the maximum number of iterations was exceeded. In this case, when the estimate is clipped onto the boundary it may not necessarily be close to any of the true roots, and hence from that position the iterative process will again be divergent.

In both of the above cases, when restriction was enforced, it was likely that the maximum number of iterations would be exceeded and another initial estimate would be chosen. It is a reasonable assumption, therefore, that if the current iterative estimate lies some distance outside the stability triangle then the process is divergent; the iterative process should be terminated immediately and a new initial estimate chosen.

As this test for divergence must be made after each iteration, a computationally-efficient test strategy must be devised. It is not necessary that the iterative estimates be restricted to within the stability triangle, and hence three options are available. The first is to test if the estimate is anywhere outside the triangle. The second and third options compare the Euclidean and Chebychev distances from the origin to the estimate with the maximum respective distance possible within the stability triangle (see Figure 4).

Table 2 shows how each test would be implemented, and Table 3 shows the respective number of comparisons, additions, and multiplications. The most computationally-efficient test is the Chebychev distance of the estimate from the origin, and hence this test was employed.

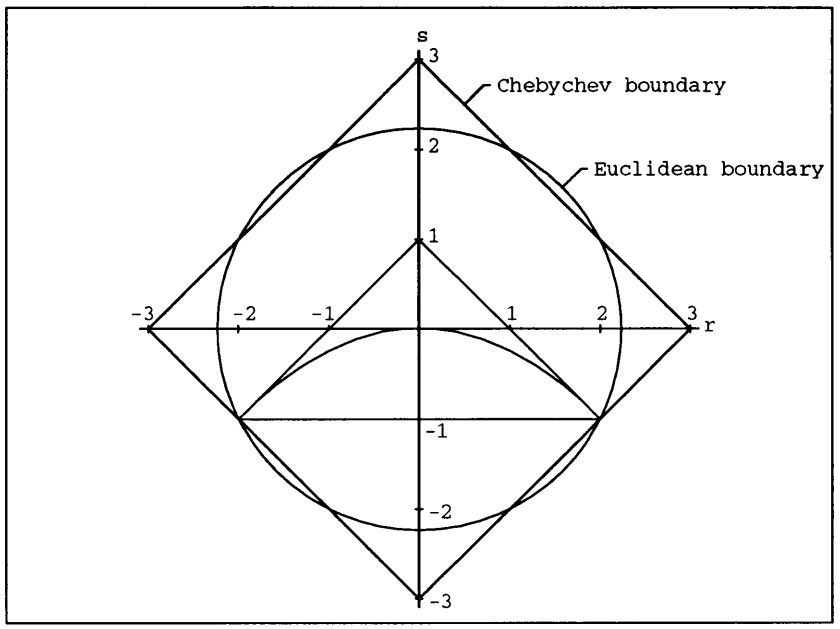

Figure 4: Boundaries for detecting divergence.

Test Criterion	Comparisons	Additions	Multiplications
outside triangle	3	2	0
Euclidean distance	1	1	2
Chebychev distance	1	1	0

Table 3: Operations required for specified tests.

A table of initial estimates was constructed from 441 points in the stability triangle, spaced 0.1 units apart in both the r and s directions. The first guess is at point $[2, -1]$, the second at $[1.9, -1]$ and so on. With the table constructed in such a manner, the first set of guesses correspond to positions on the unit circle (starting at point $1 + j0$) on the z-plane.

For slowly-varying sounds such as the vowels and semivowels, the position of the poles should vary slowly, and hence the results from the previous frame should make a good first estimate for the current analysis frame. An additional four places are reserved at the front of the estimate table for these values. For the very first frame of the utterance, these values are set equal to $[2, -1]$, $[1.9, -1]$, $[1.8, -1]$, and $[1.7, -1]$.

The word /ma/, spoken by the author, was low-pass filtered at 4 kHz cut-off (48 dB/octave fall off) and sampled at 10 kHz. The vowel was extended to give a steady-state region at the end of the utterance (see Chapter 4) and the utterance duration was 0.83 seconds. It was subsequently analysed using an 8th-order LP model, with a fixed frame size of 100 samples, with no overlap of consecutive frames, giving an analysis rate of 100 frames per second of speech. The temporal evolution of each of the LP coefficients was smoothed using a 4th-order function (see Section 3.6).

Table 4 shows the number of iterations required to determine the first 3 quadratic factors of the denominator polynomial for the first 20 frames out of 83 to an accuracy of 1 part in 100 million. In all cases, the fourth factor requires zero iterations as it is left as a quotient after the third factor. It is seen that the first frame requires more iterations than the preceding frames which take the results of the previous frame as the initial estimates. The average number of iterations required for each factor in the preceding frames is 4 or 5. This method of extracting the roots of a polynomial is therefore very well suited to this application.

3.4 The rs-Plane: an Alternative to the z-Plane

The denominator polynomial of the linear prediction model could be expressed as a multiplication of quadratic factors (see Section 3.3). These quadratics can be factorised to produce pole-pairs which could subsequently be plotted on the z-plane.

Frame No.	Iterations for		
	first factor	second factor	third factor
1	6	13	7
2	5	6	5
3	5	5	5
4	5	5	5
5	5	5	5
6	5	5	5
7	4	5	5
8	4	5	4
9	5	5	4
10	5	5	5
11	5	5	5
12	4	5	4
13	4	5	5
14	4	5	4
15	4	5	4
16	4	4	5
17	5	4	5
18	5	4	5
19	5	4	5
20	5	5	4

Table 4: Number of iterations per frame for word /ma/.

In Section 3.2, it was shown that tracking the pole movements in the z-domain has certain associated problems.

There exists a two-dimensional real space, the rs-plane (as described in Section 3.3), which is an intermediate space when moving from the multi-dimensional linear-prediction space to the single-dimensional complex space of the z-plane. The interior of the unit circle in the z-plane maps to the interior of a triangle on the rs-plane, and this mapping was used to tailor Bairstow's root-finding algorithm to the present analysis situation.

3.4.1 Properties of the rs-plane

It was desirable to track on the z-plane as the position of the poles correspond to both frequency and bandwidth of a formant. The relationship between the argument of the pole and the formant frequency is well-understood. Section 3.2 showed that tracking on the z-plane has its problems.

In this section it will be shown how lines of constant $\jmath\omega$ and constant σ on the s-plane, which are related to frequency and bandwidth respectively, map into the rs-plane. In mapping from the s-plane to the rs-plane, the z-plane is used as an intermediary, and hence the relationship between all three planes becomes apparent.

This mapping is determined in three distinct steps. First, the quadratic coefficients r and s must be given in terms of σ and ω. To determine how the lines of constant σ map onto the rs-plane, the effect of keeping σ constant in the mapping equations is then noted. Last, the lines of constant $\jmath\omega$ on the rs-plane are determined by keeping ω constant in the mapping equations.

A pair of poles with co-ordinates $\sigma \pm \jmath\omega$ in the s-domain map into the z-domain to co-ordinates $a \pm \jmath b$ where

$$a = \exp(\sigma T).\cos(\omega T) \tag{3}$$

$$b = \exp(\sigma T).\sin(\omega T) \tag{4}$$

where T is the sampling period. Combining the complex pair in the z-plane to give a quadratic:

$$(z-(a+\jmath b)).(z-(a-\jmath b)) = z^2 - 2az + (a^2+b^2)$$

$$= z^2 - rz - s$$

where r and s are given by:

$$r = 2a \tag{5}$$

$$s = -(a^2 + b^2) \tag{6}$$

Substituting equations 3 and 4 into 5 and 6 gives:

$$r = 2\exp(\sigma T)\cos(\omega T)$$

$$s = -\exp(2\sigma T)$$

For lines of constant σ on the s-plane, $\exp(\sigma T)$ in the mapping equations is a constant, M, and therefore r and s are given by:

$$r = 2M\cos(\omega T) \tag{7}$$

$$s = -M^2 \tag{8}$$

It is seen that s is a constant for constant σ, and so the lines of constant σ in the s-domain map onto horizontal lines on the rs-plane, passing through the point $[0,-\exp(2\sigma T)]$. The $\sigma = 0$ line in the s-domain therefore maps onto the $s = -1$ line on the rs-plane.

As σ tends towards $-\infty$ the corresponding line on the rs-plane tends towards $s = 0$. The end points of the lines correspond to substituting $\omega T = 0$ and $\omega T = \pi$ into the above equations, giving end points of $[2M,-M^2]$ and $[-2M,-M^2]$. Figure 5 shows five such lines for σT ranging from -1 to 0 in steps of 0.25.

For lines of constant ω, $2\cos(\omega T)$ will be a constant, K, hence $r = KM$. It can also be shown that:

$$s = -r^2/K^2 \tag{9}$$

From equation 9, it is seen that lines of constant ω map onto parabolas in the rs-plane. For the stable region of the s-plane, i.e. $\sigma \leq 0$, lines of constant ω map onto segments of these parabolas with end points [0,0] and [K,-1].

The range of ω is from 0 to πf_s, where $f_s = 1/T$ is the sampling frequency. The range of ωT is therefore from 0 to π giving a range of K from 2 to -2. This gives a family of parabolic segments, a selection of which is shown in Figure 5.

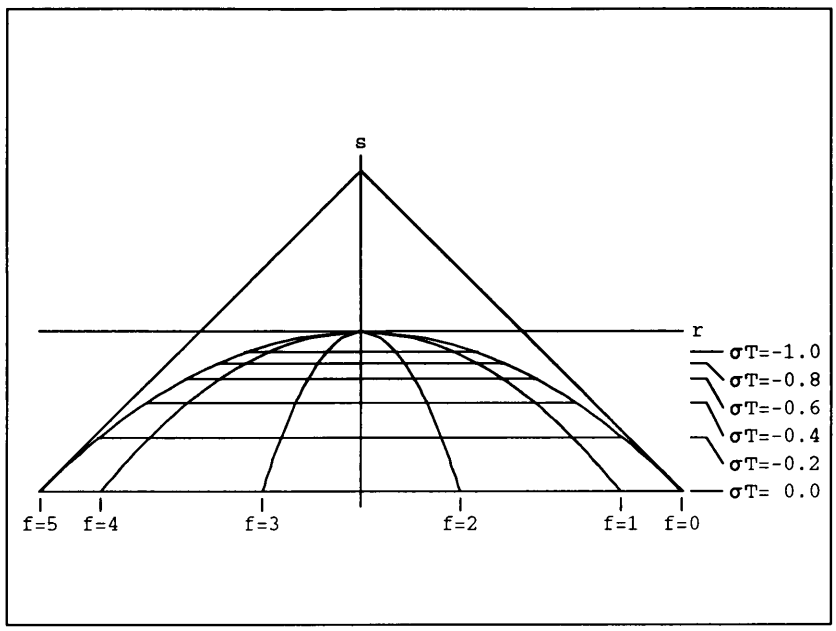

Figure 5: Lines of constant $f = \omega T/2\pi$ and σT projected onto the rs-plane.

As can be seen from Figure 5, the mapping from the s-plane, or z-plane, to the rs-plane is not conformal as the lines of constant σ and ω do not always cross at right angles. The mapping is also non-unique, as two points on the complex plane are mapped to one on the rs-plane.

These properties are generally undesirable. However, they can be exploited to solve the problems associated with tracking in the z-plane.

3.4.2 Tracking on the rs-plane

One problem associated with tracking the poles on the z-plane occurred when complex pole-pairs moved onto the real axis to become real pairs, and vice versa (see Section 3.2). Poles moving in such a fashion can be thought of as moving from a two-dimensional Euclidean space to a one-dimensional space. The discontinuity involved gives rise to problems in the correlation process.

If tracking is done in the rs-plane, there is no such problem. The regions which represent real and complex pole-pairs are both two-dimensional and contiguous. Hence there is no discontinuity in trajectory when the track moves from one region to the

other.

As was shown, the symmetry of the z-plane could not easily be utilised to reduce the number of permutations considered in the tracking process. However, since the pole-pairs in the z-domain are represented as one point on the rs-plane, a model of order p can be represented as $p/2$ parameters on the rs-plane. The number of permutations of associations is therefore reduced from $p!$ to $(p/2)!$, and the number of distances that must be calculated is reduced from p^2 to $(p/2)^2$.

Tracking in the rs-plane rather than the z-plane is therefore desirable for two reasons. The first is the elimination of the problem due to complex pole-pairs moving towards and joining the real axis to become real pole-pairs. The second is the reduction in the number of tracks which greatly reduces the amount of computation required.

3.5 Effect of Frame Overlap on Tracking

For steady-state sounds, such as long vowels, it would be expected that the LP coefficients for consecutive frames would show a high degree of correlation. However, it will be shown that the amount of overlap between consecutive frames influences the similarity of the two sets of coefficients.

The amount of such overlap depends on two factors, namely the distance between the beginning of consecutive frames and the length of the analysis frame. If the distance between the start of consecutive frames, or *frame shift*, is a fraction of the length of the analysis frames then the frames will overlap, and a proportion of the spectral qualities of one frame will be passed over into the next frame.

The variations in the coefficients may be due to two factors. Firstly, for natural speech, no two sections of speech will be exactly the same. Secondly, the LP model breaks down at the pitch pulse; subsequently errors are introduced by the pitch pulses.

For slowly-varying sounds such as the semivowels and glides, it would be expected that the coefficients would move smoothly from one state to another as the analysis frame moves through the speech segment. Again the degree of overlap of the analysis frames influences the *smoothness* of the change-over.

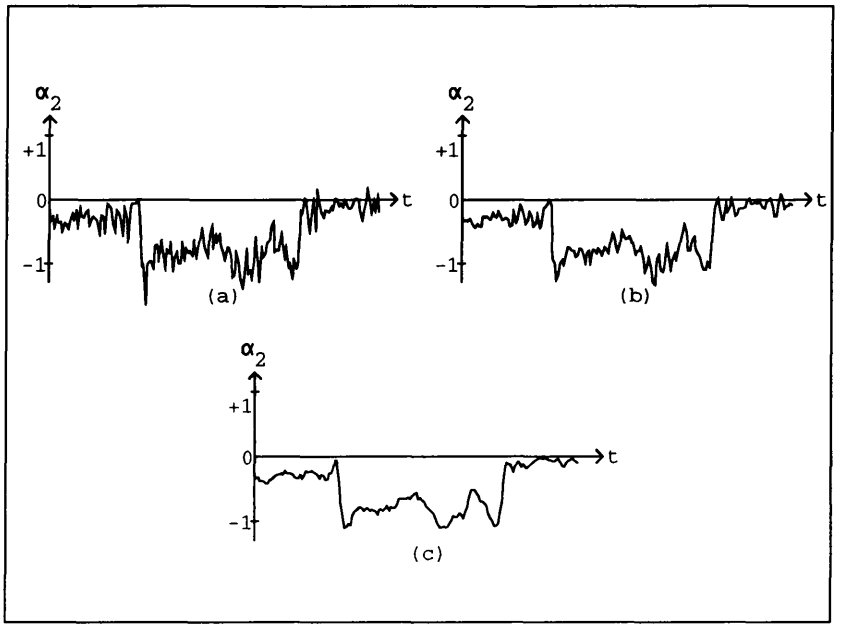

Figure 6: α_2 versus time for varying frame sizes of (a) 100, (b) 200, and (c) 500.

To investigate the effect of frame overlap on the temporal evolution of the LP coefficients, and subsequently on the tracking of the poles, the following experiment was carried out.

The word /mam/, spoken by the author with both the nasal continuant and the vowel extended, was sampled at 10 kHz and analysed using an 8th-order LP model. The frame shift was fixed at 100 samples, which gives an analysis rate of 100 frames per second of speech, and the frame size was varied. It is important to keep the frame shift constant as this ensures that the temporal scaling of each plot will be consistent.

Figure 6 shows graphs of predictor coefficient α_2 against time. Figures 6(a), 6(b), and 6(c) show the plots for frame sizes of 100, 200, and 500 samples respectively. Although the results for the 100- and 200-sample frames are noisier, they have the same basic waveshape as the 500-sample frame. It should be possible, therefore, to smooth the smaller frame size results to give an approximation to the larger frame size result.

Figure 7 shows the effect of frame overlap on the tracks on the rs-plane. The tracks shown are for a section of 35 frames which encompasses the first consonant–vowel (CV) boundary. It can also be seen that the transitions from the continuant

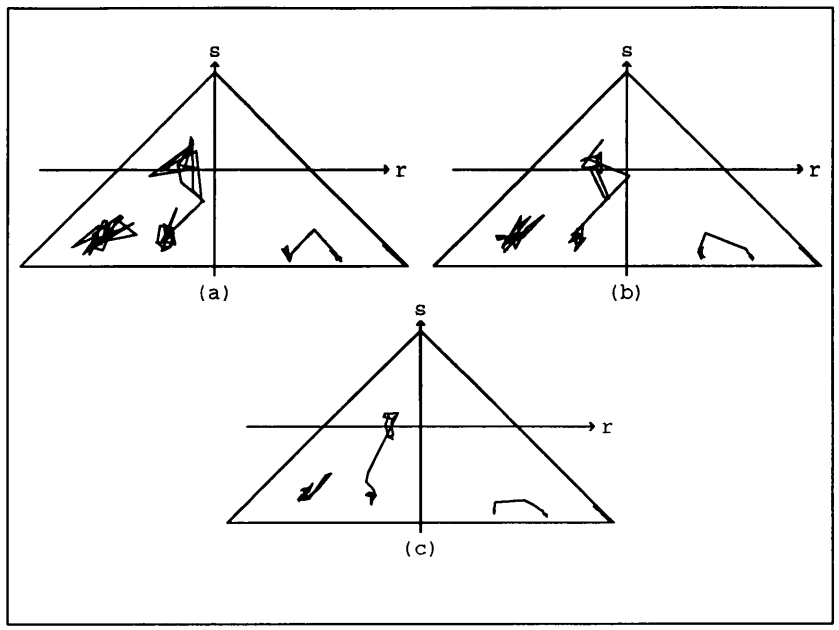

Figure 7: Section of tracks on rs-plane for varying frame sizes of (a) 100, (b) 200, and (c) 500.

consonant to vowel, and vice-versa, do not cause the LP model to go unstable, as all of the pole-pairs are within the stability triangle at all times. For this reason, such transitions are henceforth considered to be slowly-varying.

3.6 Generating Smoother Tracks

It has been shown that the LP coefficients become less variable as the size of the analysis frame is increased while keeping the distance between consecutive frames constant. There is, however, a large computational disadvantage involved when generating the covariance matrix for the LP analysis.

Figure 6 showed how one of the LP coefficients altered as the analysis frame proceeded through an utterance for three different frame sizes. Although the results for the 100- and 200-sample frame sizes are noisier, they follow the same general trend as for the 500-sample frame size. A suitable smoothing function which should approximate the 100-sample frames size case to the 500-sample frame size case, would be to low-pass filter the traces for the coefficients.

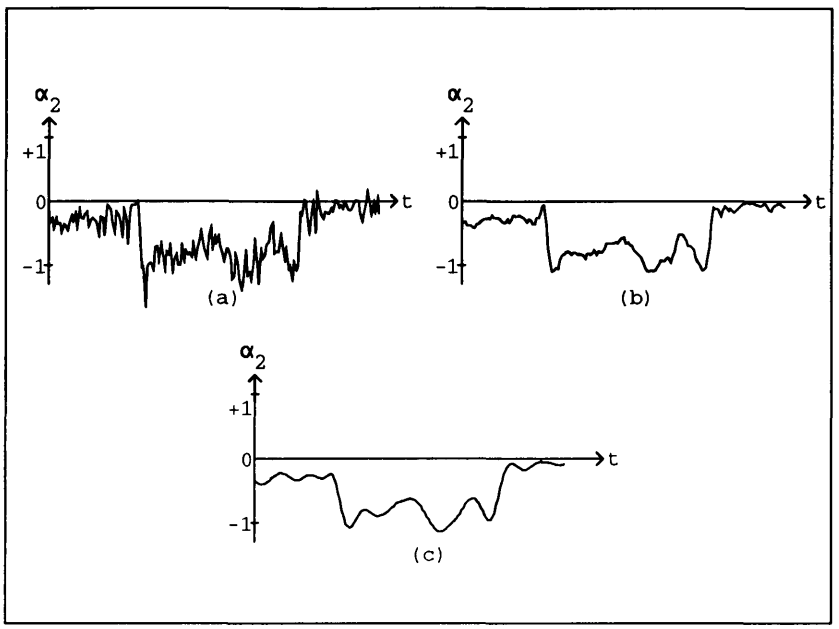

Figure 8: α_2 versus time for varying frame sizes of (a) 100, and (b) 500, and for a filtered version of frame size (c) 100.

Figure 8 shows α_2 for frames sizes (a) 100 and (b) 500, and the results of (a) smoothed using a low-pass filter. The low-pass filter process is a 4th-order maximally flat digital IIR filter with unity gain. To ease the comparison of the traces, the time delay associated with digital filters, and the lag of the filter have been manually compensated for. It can be seen that trace (c) does indeed compare favorably with trace (b), although it is derived from (a).

Considering the very complex relationship between the poles of a polynomial and its coefficients, concern may arise about the validity of the poles resulting from a polynomial which has its coefficients generated by manipulating the coefficients of other polynomials. Figure 9 illustrates that pole-pairs obtained from such a situation can indeed be valid: it shows sections of tracks taken across the CV boundary for the word /mam/ for frame sizes of (a) 100 and (b) 500, both with raw coefficients, and for a filtered version (c) of the 100-sample frame size. The tracks for (c) are good representations of those for (b) and exhibit tighter grouping of the pole-pairs during steady-state regions.

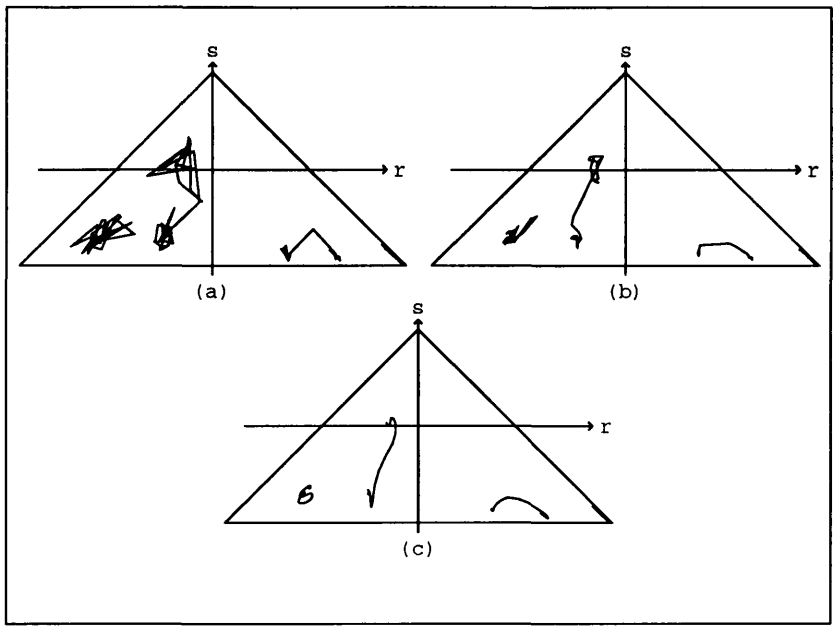

Figure 9: Tracks on rs-plane for frame sizes of (a) 100, and (b) 500, and for a filtered version of frame size (c) 100.

3.7 Conclusions

Tracking the pole movements has been proposed as an alternative to conventional formant tracking. It has advantages in that the tracking algorithm is straightforward due to the constant number of objects that are tracked throughout the speech sample: no information is discarded and close formants do not cause a problem.

The method of determining the poles of the LP model employs Bairstow's method with modifications which utilise properties of the speech signal. It is assumed that the LP model will be stable, which is reasonable for slowly varying sounds such as vowels, semivowels and transitions between continuants. This, of course, restricts the application to slowly-varying sounds.

Tracking pole movements on the rs-plane as opposed to the z-plane offers computational advantages and simplifies the required tracking algorithm. The non-unique mapping from the z-plane to the rs-plane is exploited to reduce the number of permutations considered during the tracking process from $p!$ to $(p/2)!$. For model orders

of 8 and 10, this reduces the computational load by factors of 1680 and 30240 respectively. The mapping from the z- to rs-plane is also not a conformal mapping, and this property is exploited to remove the discontinuities associated with complex pole-pairs turning real and vice-versa.

For effective tracking, it has been shown that the variations of the LP coefficients should be smooth with respect to time. Although this can be accomplished with large frame overlaps, a more effective method is to apply a low-pass filter to the evolutions of each of the LP coefficients.

3.8 References

[1] J.L. Flanagan (1954) "Automatic extraction of formant frequencies from continuous speech", *Journal of the Acoustical Society of America*, **28**, 110–118.

[2] J.D. Markel (1972) "Digital inverse filtering–a new tool for formant trajectory estimation", *IEEE Transactions on Audio and Electroacoustics*, **AU-20**, 129–137.

[3] R.W. Schafer and L.R. Rabiner (1970) "System for automatic formant analysis of voiced speech", *Journal of the Acoustical Society of America*, **47**, 634–648.

[4] A. Crowe and M.A. Jack (1987) "Globally optimising formant tracker using generalised centroids", *Electronics Letters*, **23**, 1019–1020.

[5] S.S. McCandless (1974) "An algorithm for formant extraction using linear prediction spectra", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **ASSP-22**, 135–141.

[6] S. Seneff (1976) "Modifications to formant tracking algorithm of April 1974", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **ASSP-24**, 192–193.

[7] T. Güngen and N Geçkinli (1984) "An algorithm for formant tracking", *Signal Processing*, **6**, 293–300.

[8] G. Rigoll (1986) "A new algorithm for estimation of formant trajectories directly from the speech signal based on an extended kalman-filter", *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '86, Tokyo, Japan*, **2**, 1229–1232.

[9] J.P. Olive (1971) "Automatic formant tracking by a Newton-Raphson technique", *Journal of the Acoustical Society of America*, **50**, 661–670.

[10] J.D. Markel and A.H. Gray (1976) *Linear Prediction of Speech*, 164–189, Springer Verlag, Berlin.

[11] S. Magowan (1965) "A method of plot-to-track correlation", *RRE Memo. No. 2152*, Royal Radar Establishment, Malvern, England.

[12] M.J. Maron (1986) *Numerical Analysis: A Practical Approach (2nd Ed.)*, Collier Macmillan, London.

[13] L.F. Willems (1987) "Robust formant analysis for speech synthesis applications", *Proceedings of European Conference on Speech Technology, Edinburgh*, **1**, 250–253.

[14] M. Bellanger (1984) *Digital Processing of Signals: Theory and Practice*, page 169, John Wiley & Sons Ltd. Translated into English by J. McMullan.

[15] J.G. Proakis and D.G. Manolakis (1988) *Introduction to Digital Signal Processing*, page 207, Collier Macmillan, London.

[16] J.M. Turnbull, A.T. Sapeluk and R.I. Damper (1989) "A new method of pole-tracking with application to vowel and semivowel recognition", *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '89, Glasgow, Scotland*, **1**, 569–571.

Chapter 4

Non-Real-Time Simulation

Overview

This chapter discusses the production of a non-real-time simulation of a minimum system for the speech therapy Tutor.

In addition to the parameter-tracking technique discussed in Chapter 3, the minimum system requires an automatic utterance-detection routine. This chapter discusses a method of determining the start and finish of an utterance, and for detecting any steady-state regions within the utterance.

Also covered by this chapter is the addition of a template-matching procedure to complete the simulation of a minimum system for the Tutor.

The conclusion drawn from running the simulation on sampled utterances is that the Tutor will be capable of extracting extended vowels from a CV or CVC context, and estimating the quality of the vowel.

4.1 Introduction

The speech therapy Tutor will deal with the correction of vowels which could be spoken as an isolated utterance or occur in a CV or CVC context. The vowels must be extended allowing the articulators of the vocal tract to reach a steady-state. The Tutor will then estimate the quality of the extracted vowel. To verify that it will be able to meet these specifications, it is necessary to produce a simulation of the Tutor.

In this chapter, the modules incorporated into the simulation will be discussed, the specifications for which form the basis for a minimum system, i.e. the production of a tool which is useful to speech therapists but still has room for improvement.

The philosophy behind this is that once a working real-time system has been produced, the speech therapists can evaluate the system and formulate ideas about increased functionality, enhancements to the user interface etc. In parallel with this, research into improving technical aspects, such as the template-matching process, will be carried out.

To produce the simulation, in addition to the processing steps to the tracking process described in Chapter 3, an automatic-segmentation procedure must be developed, capable of extracting an extended vowel from a monosyllabic utterance.

One method of achieving this is to track the analysis parameters and detect when they deviate (from frame-to-frame) by less than a predetermined amount. This should also ensure that the results of any analysis during co-articulation are disregarded during the final template match.

The template-matching procedure must compare the resultant pattern of parameters with a target template. This process is exactly the same as the matching process in the tracking procedure.

The full simulation of the speech therapy Tutor was written in Pascal to run on a 6 MHz IBM PC AT. Without the addition of any dedicated signal processors or auxiliary processors, such hardware is incapable of carrying out the analysis in real time.

The simulation includes all the analysis processes which will be carried out by the Tutor, excluding the perceptual scaling that is described at the end of this chapter.

Also excluded from the simulation was the display of the results in a form suitable for the patient. The display that is implemented, however, shows pole tracks on the rs-plane and gives written messages as to the closeness of match to stored templates and quality of signal with respect to background noise.

As well as detecting the start and end points of the extended vowel within an utterance, it is necessary to detect automatically the start and end point of the utterance.

4.2 LP Analysis

The LP analysis routine used in the simulation is a Pascal implementation of the covariance method of performing LP analysis (a good description of this technique can be found in [1]). This routine was written by the author and was verified by comparing its results (the LP coefficients) with that of the proprietary signal processing package Sig [2], version 1.2, which was available on the Institute's VAX cluster. Both LP analysis tools were applied to the same data sets.

The ability of the LP technique to model the spectral features of the data sets was also verified using a 1024-point fast-fourier transform (FFT) routine from the Turbo-Lader Complex package [3], version 2.0, which is available for PC compatibles. This is achieved by performing an FFT on a 1024 sample data set. LP analysis of the same data set is then carried out (using, say, a 10-th model order). The LP coefficients are the parameters of a time-varying all-pole digital filter. The impulse-response of this filter is determined over a 1024 sample period (this is normally long enough to allow the response to decay to zero). The resultant data set is then applied to the FFT routine to give the LP spectrum. A plot of the magnitude of both the FFT results shows the LP spectrum to be a smoothed version of the FFT of the original set with very similar overall spectral features.

4.3 Start- and End-Point Detection

The start and end point of an utterance can be estimated by comparing the incoming signal energy with a predetermined threshold, E_t. This threshold is chosen such that background noise is taken to be silence. The signal energy is approximated using the short-time signal energy function [1] $E(n)$, given by:

$$E(n) = \sum_{i=0}^{N-1} s(n+i)^2$$

where $s()$ is the sampled speech signal, N is the number of samples over which the energy function is calculated, and n indicates where the first sample of the frame occurs in the data. In this case, N is chosen to be the size of an analysis frame.

To estimate a suitable value for the energy threshold for this application, $E(n)$ was calculated for some background noise and E_t was taken to be ten times this value.

The start and end point of the utterance were determined as follows:

- If the signal energy is greater than the threshold energy, E_t, for 5 consecutive frames, then the first sample in the 6th frame is taken to be the start of the utterance.
- When the signal energy subsequently falls below the energy threshold, the end point of the utterance is taken to be at the end of the preceding frame.

The start is chosen to be five frames into the utterance for robustness. This ensures that short bursts of noise, such as a door closing, would be disregarded.

Figure 10 shows where this algorithm chooses the start and finish of the word /mo/ spoken by the author with the vowel extended. This figure also shows where the start and finish of the steady-state region of the utterance are chosen. The algorithm employed is discussed in the following section.

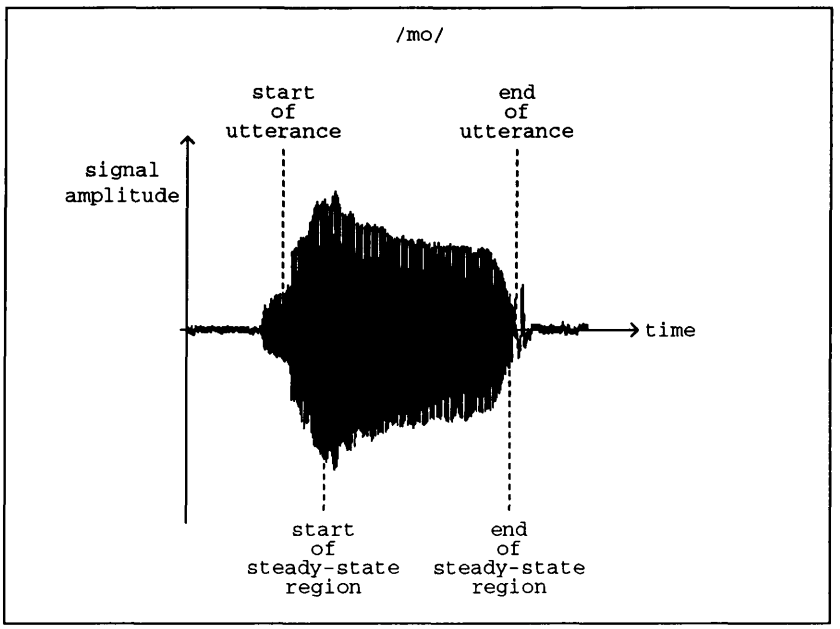

Figure 10: Key points of an utterance.

4.4 Detecting Steady-State Regions

If all of the articulators in the vocal tract are stationary during an utterance, then the analysis of the sound produced should generate consistent results from frame-to-frame. This is the condition expected during the analysis of an extended vowel.

For this application, the parameters tracked are the pole-pairs on the rs-plane. During the production of a steady-state sound, the pole-pairs should deviate only slightly from frame-to-frame. As the parameters are tracked, during the correlation process the permutation with the smallest cumulative distance is chosen. This distance is a measure of how much the parameters change from frame-to-frame and will henceforth be referred to as the *deviation*, δ. It will be shown that it is possible to use this measure to identify steady-state regions in an utterance.

Figure 11 shows δ plotted against time for the word /mam/. Note, however, that in this instance, both the nasal continuants and the vowel were extended. The two major peaks correspond to the boundary regions either side of the vowel and therefore show when the articulators in the vocal tract are moving rapidly.

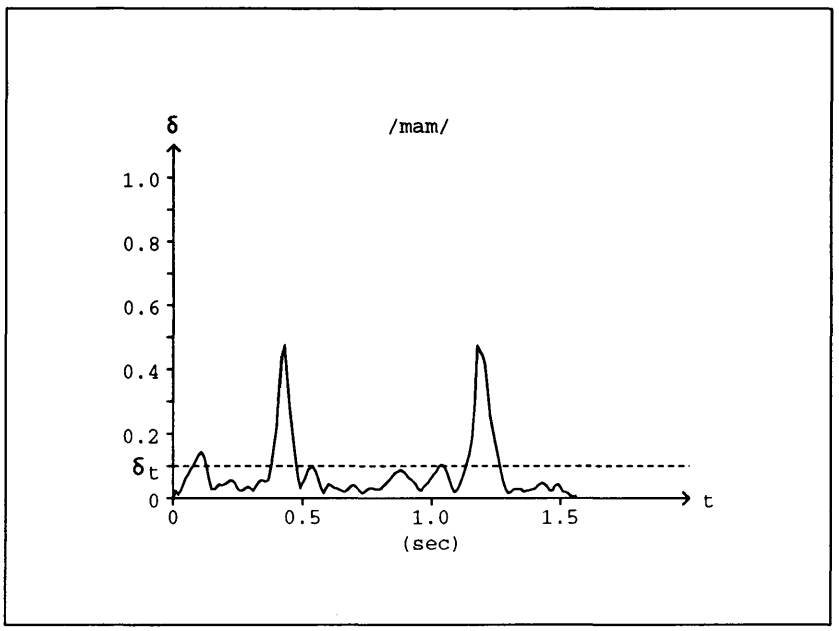

Figure 11: Deviation, δ, against time for /mam/.

Using a similar strategy to that of detecting the start and end point of the utterance, this graph can be used to estimate the steady-state regions. A deviation threshold, δ_t, is chosen by observing the deviation during known steady-state sounds. δ_t is shown on Figure 11 as a horizontal dotted line. During the analysis of an utterance, if the deviation falls below δ_t for 5 consecutive frames, the first sample in the 6th frame is taken to be the start of the steady-state region. The end of the steady-state region is marked by the frame prior to that during which the deviation rises above the threshold.

Figure 10 shows where this algorithm chose the start and finish of the steady-state region for the word /mo/.

4.5 Template for the Steady-State Region

Once a steady-state region in an utterance has been identified, it should be possible to produce a pattern of parameters which represent this sound. This could then be matched against a reference, or template.

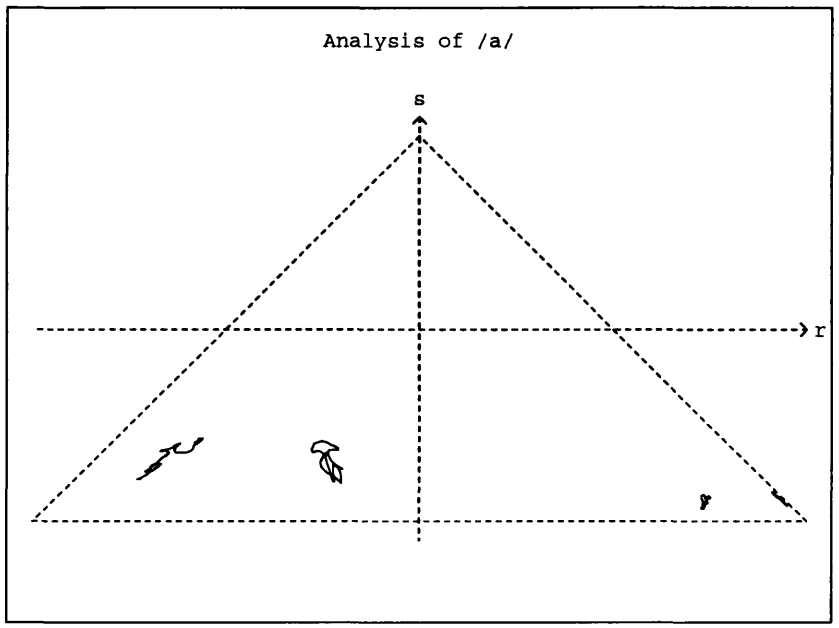

Figure 12: 8th-order analysis of /a/.

During a steady-state region of an utterance, the pole-pairs tend to vary about a mean point or move very slowly. Figure 12 shows the 8th-order analysis of the vowel /a/ which was sustained for approximately 1 second.

The low-frequency pole-pairs appear as tightly-packed traces towards the right, and hence show less movement throughout the utterance than do the higher-frequency pole-pairs. The highest-frequency pole-pair is an example of a slowly-moving track, although the direction of this movement cannot be determined from this plot, as the temporal information is lost. The next-highest-frequency pole-pair is an example of a track which moves about a mean position.

One measure of how static a pole-pair is in a steady-state region is its standard deviation over this time period. Consider Figure 13 which shows a statistical analysis of the tracks shown in Figure 12. The circles are centered about the mean position of a track and have a radius equal to the standard deviation of the track.

By inspection, it appears that two factors determine how static a pole-pair will be in a steady-state region. First, the further a pole-pair is from the $s = -1$ line (corresponding to an increasingly higher bandwidth formant) then the less static it is. Second, the less r is for the pole-pair (corresponding to an increasingly higher

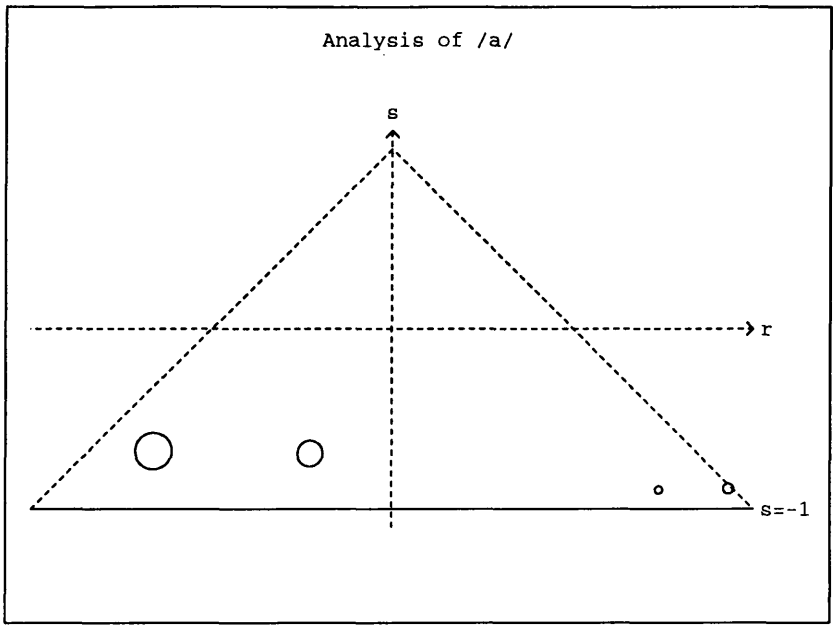

Figure 13: Standard deviations of pole-pairs.

frequency formant) then the less static it is.

To take into account this variability of pole-pairs during a static sound, a *simple* template would be the mean and standard deviation for each of the pole-pair tracks. The higher the standard deviation of the pole-pair the less significant it will be in the template match.

This template-matching method was developed to complete the simulation of the minimum system for the Tutor to enable estimates of the feasibility of real-time operation to be made. This method was originally employed by the Tutor but has been superceded by a superior technique which takes into account perceptual considerations. Both methods are described in the remainder of this chapter.

4.6 Simple Distance Measure

As was described in the previous section, the template comprises the mean and standard deviation of each of the pole-pairs during a steady-state sound. To match the pattern for an utterance with a template, the best correlation between them must be found in exactly the same way as the tracking process correlates the patterns from

frame-to-frame.

However, to account for some pole-pairs having higher deviations than others the cumulative distance metric is not used as a measure of closeness. In this case, the Euclidean distance between each pole-pair and its correlate is calculated, and this distance is divided by the standard deviation of the pair which belongs to the model template. This gives a higher weighting to distances between pole-pairs which have a lower standard deviation. The sum of these distances will henceforth be referred to as the normalised cumulative distance, D_n.

Figures 14 and 15 show the steady-state sections of the words /ma/ and /mo/, spoken by the author with the vowels prolonged, matched to a template for vowel /a/. The model template pole-pairs are designated by a cross with a circle whose center marks the mean position of the pole-pair and whose radius is the standard deviation of the pole-pair. The pole-pairs of the analysed utterance are designated in the same way with the circle alone. The correlates from template to template are connected by a straight line.

It can be seen in the case of /ma/ that the match is very close compared to that for /mo/. The normalised cumulative distances, D_n, are 2.63 and 59.88 respectively. This encouraging result offers great promise that D_n can be used as a measure of closeness of two sounds.

As will be shown in Chapter 5, the addition of white noise has a detrimental effect on the performance of the system, the extent of which depends on the order of analysis. To detect automatically possible system degradation due to low signal levels, a system which monitors the short-time signal energy could be incorporated.

One method of monitoring would be to compare the average short-time energy throughout the utterance with some predetermined threshold. If it fell below this threshold, then the utterance would be rejected on the grounds that the analysis could be unreliable. However, bearing in mind that the signal energy varies with the square of the signal amplitude, a situation could arise where the signal energy of a small number of frames was sufficiently high to raise the average above the threshold although the energy for the majority of the frames lies below the threshold.

An alternative method would be to record the percentage of frames, p, in the

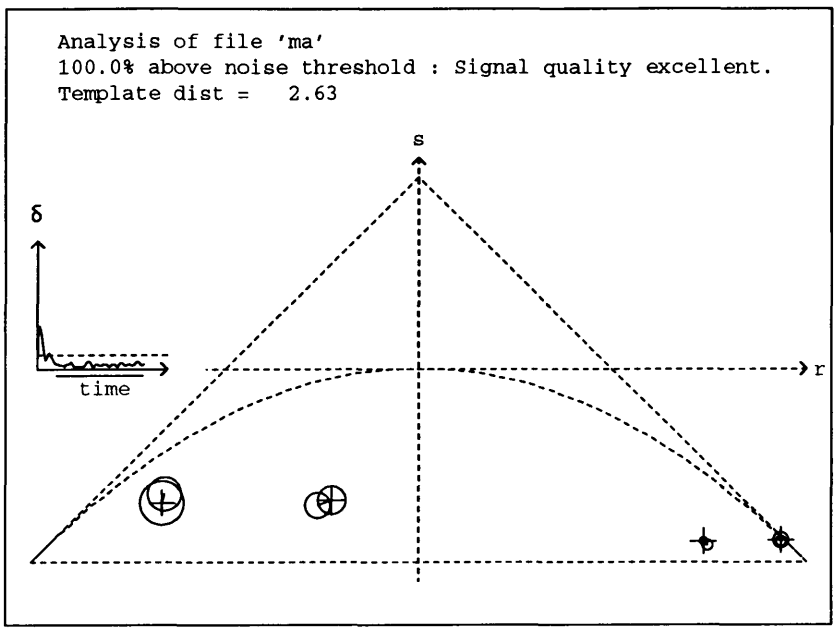

Figure 14: Template match for word /ma/ against /a/.

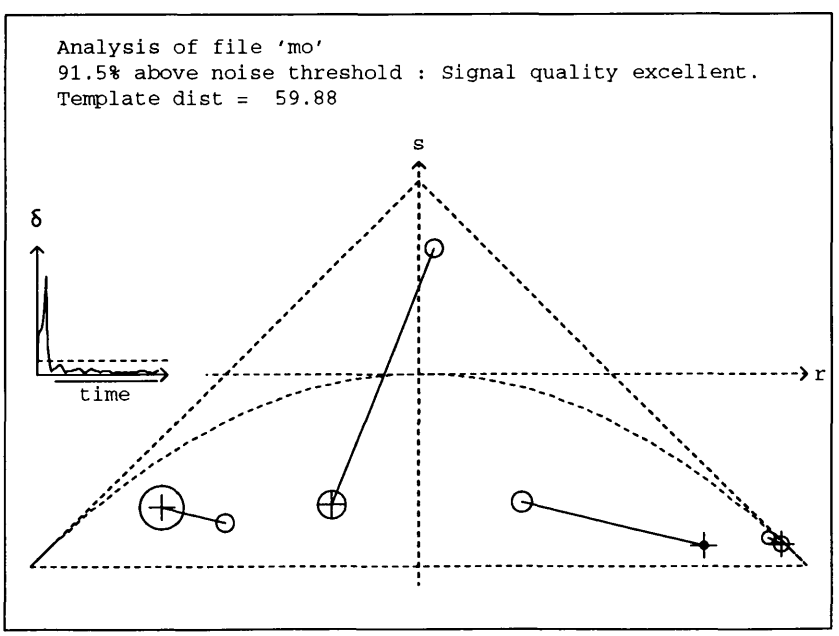

Figure 15: Template match for word /mo/ against /a/.

percentage of frames above threshold, p	signal quality
$p < 50$	bad
$50 \leq p < 60$	unreliable
$60 \leq p < 70$	fair
$70 \leq p < 80$	good
$p > 80$	excellent

Table 5: Ranges of signal quality.

utterance whose short-time energy was above the threshold. This method was felt to be superior, and was implemented in the simulation. An arbitrary threshold of 30 times E_t was chosen.

Table 5 shows the ranges used to label the *signal* quality. The signal qualities for the utterances /ma/ and /mo/ are shown on Figures 14 and 15 respectively.

4.7 Perceptual Considerations

The frequency distortion of the rs-plane gives very poor spectral resolution in the perceptually important low-frequency region. If the template matching were restricted to distances measured on this plane, then that between low-frequency pole-pairs on the target template and the utterance pattern may incorrectly be taken to be insignificant.

A more acceptable situation might be to scale the $s = -1$ (real frequency) line to make it proportional to a tonotopical scale such as the Bark scale. This would be no problem if only the region corresponding to the complex pole-pairs were considered (the relationship between points in this region and equivalent frequency is covered in Section 3.4.1). However, since the real pole-pairs can be a combination of high-frequency (half the sampling-frequency) and zero-frequency quantities there can be no such perceptual mapping method for them. The problem is to devise a mapping process which has the following properties:

- the complex pole-pairs on the rs-plane are mapped onto a tonotopical scale;

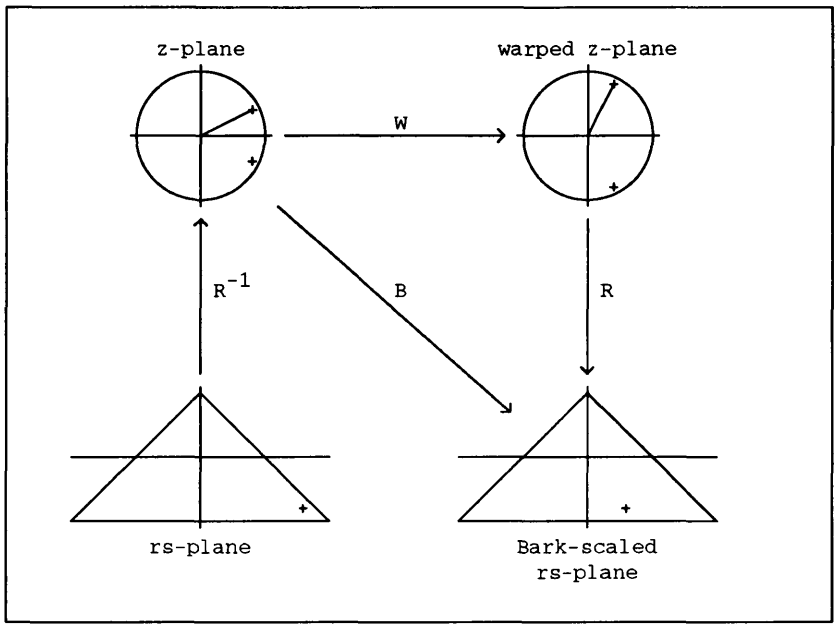

Figure 16: Steps involved in perceptually mapping the rs-plane.

- the region of the real pole-pairs remains intact and unaltered;
- the boundary between the complex and real pole-pair regions is continuous.

One solution to this problem can be found by considering the more familiar z-plane. The linear frequency scale ($\text{Arg}(z) \propto f$) is distorted when mapped onto the rs-plane. The argument of z could be pre-warped prior to mapping onto the rs-plane in such a manner as to cancel out this distortion or even distort it further so that it is proportional to a tonotopical scale.

The steps involved in this process are illustrated in Figure 16. Firstly, the vector $\vec{z}$ representing the magnitude and argument of the z-plane complex-conjugate pole-pairs must be determined from the point, $[r, s]$, on the rs-plane using the inverse of the mapping function R. Then the warping function W is used to rotate vector $\vec{z}$ about the origin, pre-warping its equivalent frequency, to give $\vec{z'}$. Lastly, $\vec{z'}$ is mapped using the function R onto the point $[r', s']$ on the perceptually-scaled rs-plane.

To determine a suitable warping function, it is necessary to consider the whole region occupied by complex-conjugate pole-pairs in both the z- and the rs-plane. However, it was shown in Section 3.4.1 that frequency measurements along lines of

constant s are proportional to the same measurements on the real-frequency line $s = -1$: equation 8 on page 26 shows this constant of proportionality to be M. This is also true for the z-plane: frequency measurements are taken from an arc with its centre at the origin, and are therefore proportional to the same distance taken along the real-frequency line on the unit circle. In fact, for both the z-plane and the rs-plane, the constant of proportionality is the magnitude of the vector $\vec{z}$ on the z-plane, M. Therefore, to determine the warping function W, it is only necessary to consider the real frequency regions, i.e. the unit circle on the z-plane and the $s = -1$ line on the rs-plane.

If r_n lies on the line $s = -1$ then the relationship between r_n and the argument, θ, of the unit length vector $\vec{z_n}$ on the z-plane is given by $r = R(\theta)$ where:

$$R(\theta) = 2\cos(\theta)$$

Therefore θ can be obtained from r by:

$$\begin{aligned} \theta &= R^{-1}(r) \\ &= \cos^{-1}(r/2) \end{aligned}$$

Subsequent to the application of the warping function, the argument of the rotated unit vector $\vec{z_n'}$, θ' is given by $\theta' = W(\theta)$.

Working back from the desired Bark-scaled rs-plane gives W: a function B can be determined such that $r_n' = B(\theta)$, where B maps $\theta = 0$ onto $r_n' = +2$ and $\theta = \pi$ onto $r_n' = -2$ and the spacing in between is proportional to the Bark scale. For this application, the mapping function employed is the analytical expression used by Traunmüller and Lacerda [4] which gives a measurement x on the Bark scale for a frequency of f Hertz as:

$$x = (26.81 \times f)/(1960 + f) - 0.53$$

Turnbull, Sapeluk, and Damper [5] show how this expression can be used to derive the warping function W. (This derivation is explained in more detail in Appendix C.)

From Figure 16 it is seen that:

$$\begin{aligned} B(\theta) &= R(\theta') \\ &= R(W(\theta)) \end{aligned}$$

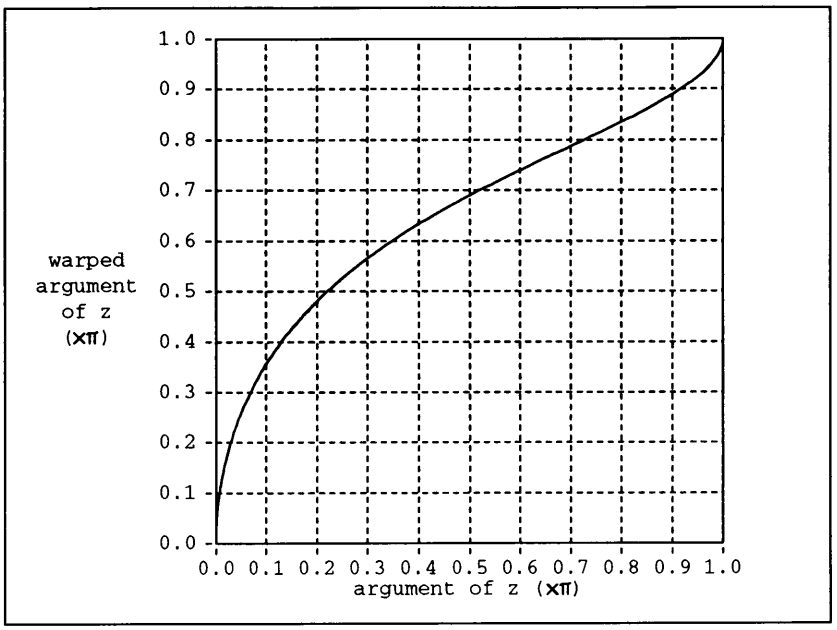

Figure 17: A plot of the warped frequency θ' against the z-plane frequency θ.

and hence

$$W(\theta) = R^{-1}(B(\theta))$$

Figure 17 shows a plot of the warped argument of z, $W(\theta)$, against the argument of z, θ. As can be seen, $\theta = \theta'$ for $\theta = 0$ and $\theta = \pi$. Hence poles on the real axis of the z-plane are not affected by the transform.

To summarise, one method of perceptually mapping the rs-plane consists of five consecutive steps:

1. calculate θ and $\vec{z}$ from the rs-plane point $[r, s]$;
2. determine the warped argument θ';
3. determine the required rotation angle, $\delta\theta = \theta' - \theta$;
4. rotate $\vec{z}$ by $\delta\theta$ to give $\vec{z'}$;
5. determine the perceptually scaled rs-plane point $[r', s']$ from $\vec{z'}$.

Implementing the transformation in this manner, though, is quite a computationally-intensive process, and is therefore quite prohibitive. Fortunately, the fact

that the magnitude of the vector $\vec{z}$ remains constant throughout can drastically reduce this computation.

It was shown in Section 3.4.1 that movements along concentric circles centred about the origin on the z-plane are equivalent to moving along lines of constant s on the rs-plane. It was also shown that the frequency divisions on these lines were all linearly proportional to the divisions on the $s = -1$ (real frequency) line. Finally, the warping function B was defined such that a point on the $s = -1$ line could be determined directly from the argument, θ, of the vector.

So to facilitate the perceptual mapping of a complex-conjugate pole-pair $[r, s]$ on the rs-plane, the following sequence of steps is required:

1. determine the point $[r_n, -1]$ at which the line of constant frequency which passes through the point $[r, s]$ cuts the $s = -1$ line using $r_n = r/M$ where $M = \sqrt{-s}$;
2. determine the argument, θ, of the equivalent vector $\vec{z}$ using $\theta = \cos^{-1}(r_n/2)$;
3. determine the perceptually weighted equivalent $[r'_n, -1]$ using $r'_n = B(\theta)$;
4. determine the point $[r', s]$ on the line of constant frequency which passes through $[r'_n, -1]$ using $r' = M \times r'_n$.

4.8 Conclusions

A procedure which is capable of detecting the start and end point of an utterance has been described.

It has been shown that the boundary between continuant sounds produces a *spike* in the plot of the distance between consecutive frames against time. This feature of the tracking process has successfully been employed as a method of extracting an extended vowel from a monosyllabic utterance.

The simple matching process that has been described should be adequate for the prototype Tutor. It has been shown that the distances resulting from matching patterns can range from the order of 2, when matching close sounds, to a distance of the order of 60, when matching distinct sounds.

However, there are deficiencies in template matching on the rs-plane. As was shown in Chapter 3, the effective frequency scale is compressed for low and high frequencies. The effect of this is to produce low-frequency pole-pairs which show only small movements parallel to the r axis.

Incorporating the standard deviation of the pole-pairs into the template-matching process might take care of this to some extent: the normalised distance increases with respect to the Euclidean distance as the standard deviation of the pole-pair decreases.

However, there is a problem associated with using such normalised distances. Since Euclidean distances are divided by the standard deviation of the reference template, the normalised distances between the template and the most recent utterance pattern is not the same as would be observed if the two were transposed. This problem has been eradicated with the alternative perceptual method.

It has been shown from running the simulation on sampled utterances that the prototype Tutor will be capable of extracting extended vowels from a CV or CVC context, and estimating the quality of the vowel.

The simulation described was implemented on an IBM PC AT, and the analysis procedures were coded in Pascal. This simulation takes approximately 25 seconds to analyse one second of speech. To achieve real-time operation, the processing power of the system must be increased by a factor of at least 25.

4.9 References

[1] L.R. Rabiner and R.W. Schafer (1978) *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ.

[2] Techni-Soft (1987) *Sig User's Manual* Livermore, CA.

[3] Lauer & Wallwitz (1986) *Turbo Lader Complex* Erlkönigweg 9, 6200 Wiesbaden.

[4] H. Traunmüller and F. Lacerda (1987) "Perceptual relativity in vowels", *Speech Communication*, **6**, 143–157.

[5] J.M. Turnbull, A.T. Sapeluk, and R.I.Damper (1990) "Pole-tracking in a vowel trainer for speech therapy", *Proceedings of the IASTED International Symposium Signal Processing and Digital Filtering*, Lugano, Switzerland.

Chapter 5

Optimal Order of the LP Model

Overview

This chapter discusses the factors considered in determining the optimal LP model order for the use in the computer-based speech therapy Tutor. Considered first is some of the vast body of work that has been carried out on the representation of vowels by just two formants or similar parameters. The question that arises from this study is "is a 4th-order LP model capable of representing a two-formant model?"

Also considered are the effects of the LP model order on the computational overheads of the parameter tracking and template-matching procedures. Finally, an experimental procedure will be discussed, which compares 4 model orders in both a relatively noise-free environment, and with the addition of white noise. The results show that an 8th-order LP model gives best performance for the application of the analysis and tracking procedures described in previous chapters.

5.1 Introduction

Previous chapters have discussed a method for tracking parameters of a speech signal in order to locate the boundaries of an extended vowel within a CVC context. Also discussed was a corresponding template-matching process which could be used to estimate the quality of the vowel. This chapter addresses the problem of obtaining the optimal order of the LP model for this application.

There are computational disadvantages in using a large model order ($p \geq 10$) with such a parameter-tracking method. Also with very small model orders ($p \leq 6$), the computational advantage of tracking on the rs-plane, rather than on the more familiar z-plane, would be slight.

It is therefore necessary to estimate the best model order for this particular application. Two conflicting ideas immediately spring to mind. On the one hand, when considering that the speech therapy Tutor is to estimate the quality of a vowel-like sound, one would be forgiven for assuming that the more information made available about the sound the better, provided it is not spurious or irrelevant as far as vowel identity goes. In this case, a high model order should be used to obtain the necessary information. On the other hand, abundant psychoacoustic evidence suggests that as few as two formants are sufficient for the identification of vowels by human listeners [1,2,3,4].

In estimating the optimal order of analysis, this chapter looks at the work done on two-formant vowels and addresses the following questions:

- what computational and physical limitations affect the upper limit of the model order?
- are two formants (or similar parameters) capable of distinguishing between two vowel-like sounds in an effort to judge vowel quality differences?
- to what extent is LP analysis capable of estimating these two formants?

5.2 Two-Formant Vowels

There has been a vast amount of work in both the fields of speech synthesis and speech recognition on reducing the identity of vowel sounds to just two parameters, and in particular, two formant frequencies.

Psychophysical experiments on the perception of synthetic vowels have shown that vowel identity can be reduced to two parameters. The first parameter is F_1. The second parameter turned out not to be F_2 but rather a function of F_2 denoted F_2'. Work has also been carried out which attempts to keep the reduced information of F_1 and F_2', but representing the parameters on a perceptually-mapped space.

5.2.1 Determining F_2' from F_2 and Higher Formants

Carlson, Granström, and Fant [1] devised a psychophysical experiment which allowed them to estimate F_2' for Swedish vowels. The validity of these estimates was then determined using a further psychophysical experiment in which participants had to identify the two-formant synthesised vowel that they heard.

Having obtained positive results, and comparing their findings with that of earlier investigations, they then tried to obtain a formal method of predicting F_2' from a weighted mean of F_2, F_3, and F_4.

Their conclusions were "...it appears difficult to extract from the four-formant vowel a frequency equal to the matched F_2', by any formalism, before the vowel is identified." In other words, F_2' could be estimated from F_2 and higher formants only if the identity of the vowel was known in advance.

Furthering their work of 1970, Carlson, Granström, and Fant [2] used an intuitive approach to establish a formula which would calculate F_2' from F_1 to F_4. The resultant equation however, contained a 'fiddle factor' which was used to improve the match of the calculated F_2' to the measurements of their particular experiment. In their conclusions, it was stated that detecting F_2' in natural speech as opposed to synthetic speech poses more of a problem. It was also suggested in their general discussion that a further parameter should be used when attempting to distinguish between voiced consonants and vowels. Such a suitable parameter would be the relative amplitude of

F_2'. For the particular application presented in this Thesis, it is necessary to identify a vowel sound within a CV or CVC context. Matching the poles on either the z-plane or the rs-plane takes into account not only frequency of the formant but a measure which relates to the bandwidth of the formant. However, is such a measure representative of the operation of the auditory system?

5.2.2 Tonotopical Distances in Two-Formant Vowels

It is widely known that the response of the peripheral auditory system to external stimuli could be grouped into *critical bands* in the frequency domain (for example [5]). A critical band defines a frequency range for which the perception of a narrow-band stimulus abruptly changes when the frequencies of the stimulus move beyond the band.

Zwicker and Terhardt [6] mapped the frequency domain onto a tonotopical scale, the Bark scale, where 1 Bark is equal to one critical bandwidth. This scale is constructed such that equal distances on this scale correspond to *perceived* equal distances between acoustic stimuli.

Traunmüller and Lacerda [7] set out to and, to some extent, succeeded in quantifying distinctive features in terms of F_1 and F_2' mapped onto the Bark scale. They also succeeded in mapping phoneme boundaries on a two-dimensional space consisting of parameters measured on the Bark scale. The boundaries obtained, however, varied depending on the native language of the subjects participating in the psychophysical experiments and, more importantly, the order of presentation of the test data to the subject.

Hermansky, Hanson, and Wakita [3] proposed a method of processing the speech signal using perceptual techniques, prior to applying a low-order LP analysis. The complete analysis technique was named perceptually-based linear prediction, or PLP. The resultant PLP spectrum was shown to be consistent with the (F_1, F_2') concept of Carlson, Granström, and Fant. The PLP technique was then applied to an automatic speech recognition problem [4], and shown to be capable of giving good results in a multi-speaker environment.

The prime disadvantage of this technique is the vast amount of processing that is

involved prior to the application of a standard LP analysis technique.

Bearing in mind that the Tutor is to be portable, it was felt by the author that the work by Traunmüller and Lacerda was the most applicable. It should be possible to estimate the formant frequencies of the speech signal and subsequently incorporate some perceptual model to map these parameters onto a tonotopical scale to improve any distance measures.

The standard method of estimating the formant frequencies of the speech signal is to obtain the poles of the LP model and then to map these discrete-time parameters into the complex-frequency domain [8]. The factors that could help determine the optimal order of the LP analysis are considered in the next section.

5.3 Effect of LP Model Order on Computation

Bladon [9] criticised the use of formant frequencies in the representation of vowels on three counts, one being the reduction aspect. His key concern is that "the discarded information would contain much that is auditorily relevant," and he presents evidence to reinforce this concern. One example of a parameter other than formant frequency is formant damping which can be used in the detection of nasality.

This scepticism about reduction agrees with instinct. One would be forgiven for feeling that to estimate the *quality* of a vowel, rather than the *identity* of the vowel, the more information one has about it the better. This instinctive approach therefore points to a large model order.

On the other hand, the processing time will obviously increase as the model order increases, and since one principal requirement of the Tutor is that it operates in real-time, this relationship must be considered further.

If the number of samples in an analysis frame, N, is much greater than the LP model order, p, then, to a reasonable approximation, the number of multiplies required to obtain the LP coefficients increases linearly with p (see Rabiner and Schafer [10]).

Calculating the computational overhead of the root-finding algorithm is impossible due to its iterative nature: if the coefficients of a polynomial are essentially random, then the number of iterations to find the roots of the polynomial is unpredictable.

However, assuming that the number of iterations remains constant, say I, then it can be shown that the number of multiplies required to obtain the roots of a polynomial of order p is $I(p^2 + 5p - 14)$ (see Appendix A).

If tracking of the poles is carried out on the z-plane then the number of parameters that must be tracked, q, is $q = p$. If the tracking is done on the rs-plane, two z-plane poles map onto a single point, thus $q = p/2$. When tracking the poles from frame-to-frame, the number of permutations that must be considered would be $q!$ (see [11]). The computation involved in each permutation depends on the distance metric used. See Appendix A for the derivation of the number of arithmetic operations. For the complete tracking process, the Chebychev metric requires $q!(q-1) + 3q^2$ additions. The square of the Euclidean distance measure requires a further $2q^2$ multiplies, and the Euclidean distance metric requires a further q^2 square-root operations. So for the simplest case, the number of additions required would be $q!(q-1) + 3q^2$.

As the Tutor is constructed with a multi-processor architecture, the analysis modules will not all be implemented on the same processor. It is therefore difficult to estimate the relative load of the modules on the complete system. However, as the processing required by the tracking algorithm grows with $q!$, it is clear that it would be dominant with large model orders.

It is obviously a major concern to keep the number of parameters tracked to a minimum. The two-formant model of Carlson, Granström, and Fant would indeed accomplish this. However, the question remains about the adequacy of the model and the best method of determining F_2'.

In an ideal situation, the number of formants that can be estimated by an LP model of order p is $p/2$. Is it possible, therefore, to estimate F_1 and F_2' with a 4th-order system? Taking the other point of view that the more information about the speech sound the better, should the largest model order that the computer-based system can handle be used?

To attempt to answer these questions, an experiment was devised that would test the steady-state sound detector with model orders varying from 4 to 10 in steps of 2.

5.4 Estimating the Optimal LP Model Order

The purpose of this experiment was to estimate the best model order for the application of the Tutor. The principal techniques under test were the tracking procedure and the steady-state detector.

It should be noted that no pattern matching was carried out in this test: the matching process requires target templates which are generated by the procedures under test, therefore whatever model order turns out to be best suited to the tracker and the steady-state detector must be best suited to the Tutor as a whole.

For the tracking and the steady-state detector to be successful there must be sufficient deviation in the pole-pair positions during the consonant–vowel boundary region of the analysed utterance. This can be shown by making a graph of the cumulative deviation of the poles against time. The individual deviations of the pole-pairs can be shown, to some extent, by plotting their tracks on the rs-plane.

Once the steady-state regions have been detected, the mean position of the pole-pairs during these regions can also be plotted on the rs-plane. With prior knowledge of the phonetic identity of the utterance, and using the plotted information on the rs-plane, it is possible to make a qualitative assessment of the separation between the patterns representing the respective phonetic elements. This will indicate whether or not that particular LP model order is capable of distinguishing between and making an estimate of the quality of these phonetic elements.

Due to the computational disadvantages of tracking on the z-plane, the tracking for this experiment was carried out on the rs-plane. It is therefore necessary to use *even* model orders throughout, as pole-pairs on the z-plane are mapped to single points on the rs-plane.

A reasonable upper limit on the order of the LP model is determined by the computational overhead in the tracking process. For real-time operation on a realistic system, the absolute maximum for the model order was estimated at 10.

Abundant psychoacoustic evidence suggests that as few as two formants are sufficient for the identification of vowels by human listeners. In ideal circumstances, each formant is modelled by 2 poles (a complex pole-pair), consequently the minimum

model order was set to 4. The model orders used in the experiment would therefore be 4, 6, 8, and 10.

5.4.1 Performance in a Quiet Environment

To enable the steady-state detection to be tested, and the pattern separation to be estimated, each test utterance consisted of two extended continuants. In each case the first phonetic element was the nasal continuant /m/, prolonged for approximately half a second. The second phonetic element was one of vowels /a/, /i:/, /o/, and /u/, also sustained for about half a second. These utterances were spoken by two male speakers, JT and DB, and low-pass filtered at 4 kHz cut-off (48 dB/octave fall-off). They were sampled at 10 kHz with 16 bits per sample. A 0.8 second section was manually extracted from the sampled utterances with the estimated phonetic boundary at approximately mid-point in the section, giving test data of consistent length and known position of the consonant–vowel boundary.

This gave 8 test input utterances for the analysis procedures, which in turn, were analysed using each of the 4 model orders (a total of 32 results). Figures 18 to 22 show typical output plots. Appendix E shows the results for the whole experiment. Figures 32 to 63 are for the low noise experiment. The remaining figures are for the added noise experiment.

Each figure shows a plot of the tracks on the rs-plane, and a graph of cumulative deviation against time in seconds. The detected steady-state regions are marked with a line just under and parallel to the time axis. Ideally, for the test utterances chosen, there should be only two such regions detected, one for the consonant, and one for the vowel.

The tracks of the pole-pair movements are shown on the rs-plane. For each steady-state region, the mean position of the pole-pairs is calculated and marked with a symbol on the diagram. The first such average position is marked with a triangle, the second with a cross, and the third with a circle. If any further steady-state regions are detected, then the fourth is again marked with a triangle, the fifth with a cross, and so on.

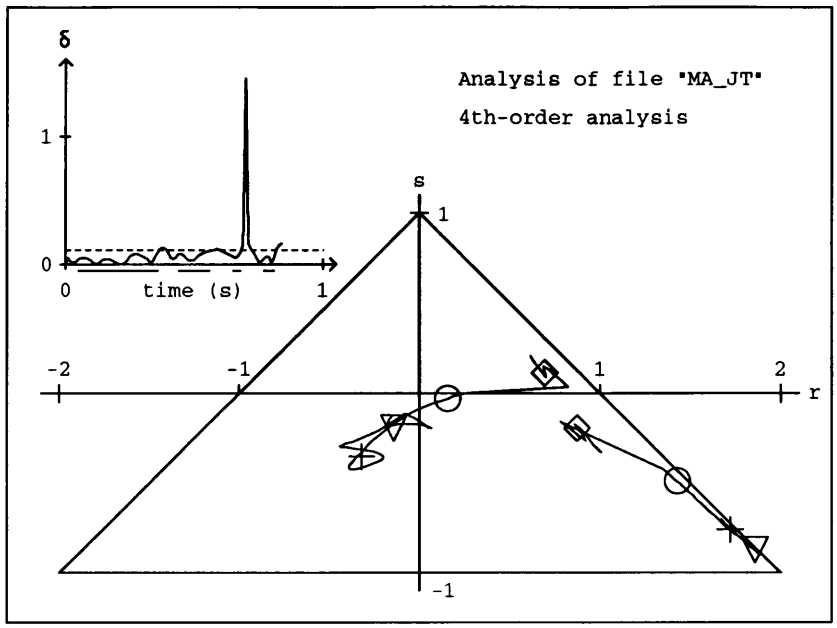

Figure 18: Analysis of /ma/ using a 4th-order model.

For a model order to be successful for a particular utterance, the following specifications must be met.

1. In the first half of the utterance (the consonant), one steady-state region must be detected which encompasses a large percentage of the segment. Other steady-state regions will be tolerated provided they are small in comparison or show roughly the same mean positions as the largest region.

2. The specifications of 1 apply to the second half of the utterance (the vowel).

3. The mean positions of the pole-pairs within the consonant are markedly different from those of the vowel.

Examining Figure 18, the result of analysing the utterance /ma/ using a 4th-order model, shows a large steady-state region during the consonant, but a very dynamic region for the vowel, which should, of course, be steady-state also. It therefore passed on point 1, but failed on point 2. As it is impossible to estimate the pattern corresponding to the vowel it must also fail on point 3.

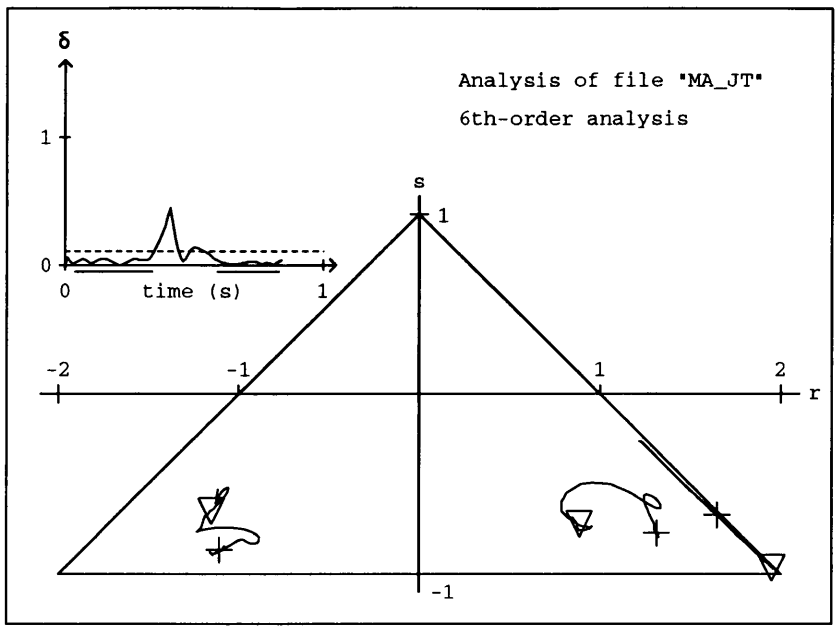

Figure 19: Analysis of /ma/ using a 6th-order model.

Figure 19 shows the result of applying a 6th-order model to exactly the same input data. There are distinct steady-state regions for both the consonant and the vowel, so it passes on points 1 and 2. There is a reasonable difference between the patterns for the two phonetic elements, therefore it passes specification 3.

Continuing the examination of all these results (see Appendix E) allows Table 6 to be constructed. The entries in the table that pass on all three counts show similar features to that shown in Figure 19: i.e. there is a distinct region near the centre of the deviation/time graph which shows a high deviation relative to the deviation on either side.

The 4th-order analysis of /mu/ (Figure 38) was incapable of distinguishing the consonant /m/ from the vowel /u/. Since the tracks on the rs-plane are reasonably tightly grouped, the patterns for both phonetic elements must be very similar.

Using a model order of 6 (Figure 46), the analysis of the utterance /mu/ showed the characteristic *spike* in the deviation/time plot. However, because the threshold for distinguishing between steady-state and dynamic sounds is too high, the procedures were unable to detect the dynamic region. Had the threshold been lower, the phonetic elements would have had patterns that were distinguishable. The 8th-order analysis

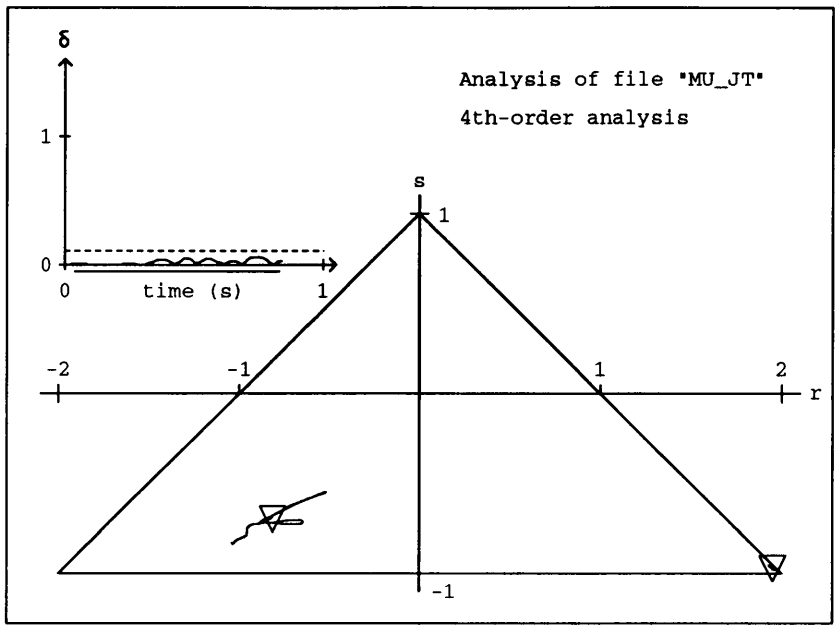

Figure 20: Analysis of /mu/ using a 4th-order model.

of the same file (Figure 54) manages to detect the steady-state region.

Would there be any justification in scaling down the threshold depending on the model order? One would expect that with fewer pole-pairs a smaller deviation from frame-to-frame would be seen. This would call for a linear scaling of the threshold depending on the model order. However, the 4th- order analysis of /mi:/ (Figure 34) and /mo/ (Figure 36) gave acceptable results with the fixed threshold. Also the 6th-order analysis of /mi:/ (Figure 42) would probably have failed to detect suitably long steady-state regions for both the phonetic elements.

The 10th-order analysis of both /ma/ (Figure 56) and /mi:/ (Figure 58) failed to obtain steady-state regions of reasonable length for the phonetic element /m/. However, for /ma/, the two short steady-state regions that were detected resulted in very similar mean positions of the pole-pairs. On the other hand, the 10th-order analysis of /mo/ (Figure 60) and /mu/ (Figure 62) did succeed in producing acceptable results for the phonetic element /m/. The analysis of /mu/ with a 10th-order model failed primarily due to the very dynamic pole-pair which has a section very close to the origin. This track was very static during the analysis of the consonant, but became very deviant during the analysis of the vowel.

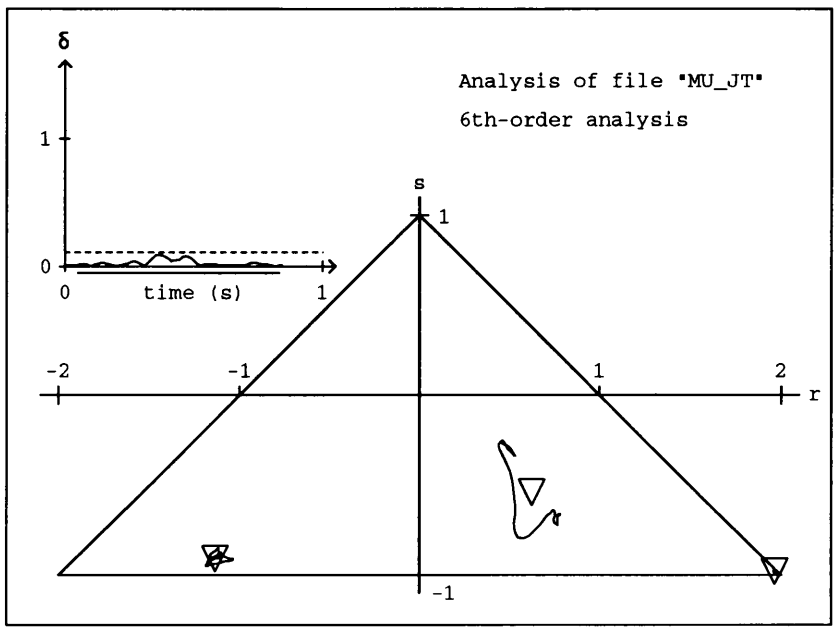

Figure 21: Analysis of /mu/ using a 6th-order model.

5.4.2 Performance with Added Noise

To establish the effect of noise on the system the previous experiment was repeated on different samples of the same utterances. In this case the samples were mixed with the output of a Gaussian noise generator prior to sampling. The estimated signal to noise ratio was 14 dB.

Table 7 shows the results of checking the output plots to see if they meet the criteria defined in the Section 5.4.1.

For the 4th-order analysis, all plots are remarkably similar (Figures 64 to 67). The pole-pairs are very static, and the whole utterance is shown as being steady-state. All the graphs of deviation against time show an increase in deviation at approximately half-way through the utterance, i.e. at the consonant–vowel boundary. Lowering the threshold would probably result in two steady-state regions being detected, one for each phonetic element. However, for all the utterances, the pole-pairs are very tightly grouped, and do not show any separation which would indicate the ability to distinguish between the consonant and the vowel.

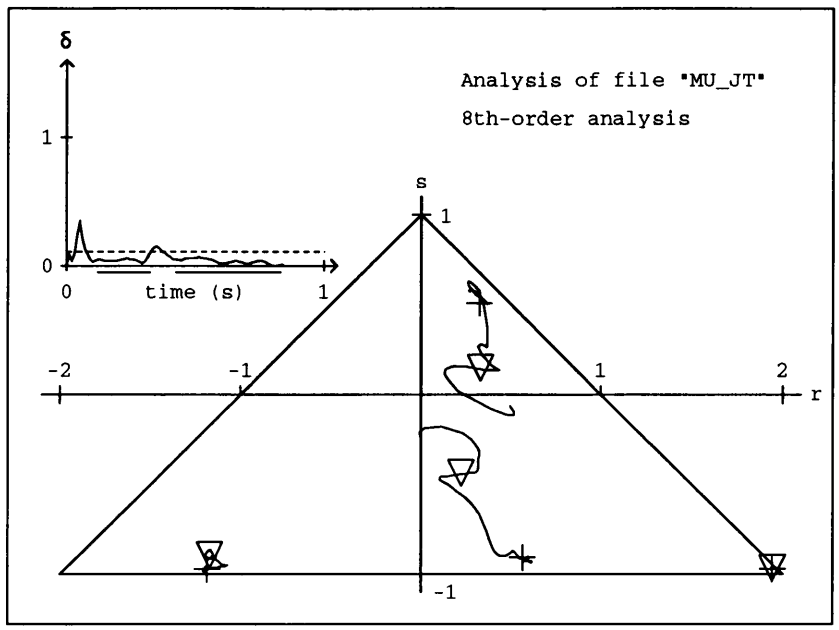

Figure 22: Analysis of /mu/ using an 8th-order model.

It should also be noted that the mean positions of the pole-pairs are almost identical for all four of the utterances. This suggests that this model order would be incapable of distinguishing between the phonetic elements /m/, /a/, /i:/, /o/, and /u/.

For the 6th-order analysis, the process fails for the utterances /ma/ (Figure 68) and /mi:/ (Figure 69). For /ma/, reasonably long steady-state regions are found for both the consonant and the vowel. During the vowel, though, two regions are found, and examining the plot shows that the mean positions of the pole-pairs are not consistent. The analysis of this utterance, therefore, fails to meet the separation criterion.

For /mi:/, if the threshold was lowered, then two steady-state regions would have been found, which have a reasonable separation between their respective patterns. Lowering the threshold, though, would have caused the length of the steady-state region in the vowel of /mo/ to be unacceptably short.

The only utterance that failed to meet all the criteria in the 8th-order analysis was /ma/ (Figure 72). However, although the results for the consonant were unsatisfactory, the steady-state region for the vowel was very long, and the tracks of the

Model order	Condition number	Utterance							
		/ma/		/mi:/		/mo/		/mu/	
		JT	DB	JT	DB	JT	DB	JT	DB
4	1	+	+	+	−	+	+	−	−
	2	−	−	+	−	+	+	−	−
	3	−	−	+	−	+	+	−	−
6	1	+	+	+	+	+	+	−	−
	2	+	+	+	+	+	+	−	−
	3	+	+	+	+	+	+	−	−
8	1	+	+	+	+	+	+	+	+
	2	+	+	+	+	+	+	+	+
	3	+	+	+	+	+	+	+	+
10	1	−	+	−	+	+	+	+	−
	2	+	−	+	+	+	+	−	−
	3	−	−	−	+	+	+	−	−

Table 6: Results of experiment in a relatively noise free environment. A + indicates a pass and a − indicates a fail.

pole-pairs during this region were very tightly grouped. Such a result would be far from disastrous for a therapy aid for *vowel* production. This argument also applies to the 10th-order analysis of the same utterance and for /mu/ (Figure 79).

Using a 10th-order model when analysing /mi:/ (Figure 77) and /mo/ (Figure 78) resulted in no steady-state regions of reasonable length being detected.

5.5 Conclusions

A 4th-order LP model does not appear to be a satisfactory equivalent of the two-formant system of Carlson, Granström, and Fant. A model order of p could represent $p/2$ formant frequencies, and hence it was hoped that a 4th-order model might represent F_1 and F'_2. However, in the majority of cases, this model order was incapable of distinguishing the consonant and the vowel of an utterance.

One possible explanation for this phenomenon can be seen from examining the 4th-order analysis of /mu/ in the *low noise* experiment with the 6th- and 8th-order

Model order	Condition number	Utterance			
		/ma/	/mi:/	/mo/	/mu/
4	1	+	+	+	+
	2	+	+	+	+
	3	−	−	−	−
6	1	+	+	+	+
	2	−	+	+	+
	3	−	−	+	+
8	1	−	+	+	+
	2	+	+	+	+
	3	−	+	+	+
10	1	−	−	−	−
	2	+	−	−	+
	3	−	−	−	−

Table 7: Results of experiment with added noise. A + indicates a pass and a − indicates a fail.

analyses of the same sample (See Figures 20 to 22). The mean positions of the pole-pairs for the vowel /u/ on the 8th-order plot, represented by crosses, show 3 pole-pairs close to the $s = -1$ line, and hence 3 narrow-band formant frequencies. The central one, with approximate co-ordinates (0.5,-0.9), is slightly further from this line and hence has a larger bandwidth. The 6th-order plot shows a similar pattern. The major difference between the patterns for the /m/ and the /u/ is a change in bandwidth of F_2. It would appear that the gross spectral representation of the 4th-order model favours the positions of the two highest amplitude formants, and hence it fails to detect the movement in the less strong one. For the added noise experiment, the effect of the noise must be sufficient to cause the *gross* spectral shape to appear constant to the 4th-order model.

The 10th-order LP model suffers from highly-deviant pole-pairs, even during steady-state sounds. One explanation for this could be that if $p/2$ is greater than the number of strong formants in the speech signal, the remaining pole-pairs will model wide spectrum noise, or the roll-off of the anti-aliasing filter.

The best results in both of the experiments were obtained with the 8th-order

model, and for this reason, this model order was chosen for the Tutor.

5.6 References

[1] R. Carlson, B. Granström, and G. Fant (1970) "Some studies concerning perception of isolated vowels", *STL-QPSR*, **2–3**, 19–35.

[2] R. Carlson, B. Granström, and G. Fant (1975) "Two formant models, pitch and vowel perception", *in* G. Fant and M.A.A. Tatham (editors), *Auditory Analysis and Perception of Speech*, 55–82, Academic, London.

[3] H. Hermansky, B. A. Hanson, and H. Wakita (1985) "Perceptually based linear predictive analysis of speech", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, ICASSP '85, Tampa, Fl*, 509–512.

[4] H. Hermansky, K. Tsuga, S Makino, and H. Wakita (1986) "Perceptually based processing in automatic speech recognition", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, ICASSP '86, Tokyo, Japan*, 1971–1974.

[5] D. O'Shaughnessy (1987) *Speech Communication: Human and Machine*, Addison-Wesley, Chapter 4.

[6] E. Zwicker and E. Terhardt (1980) "Analytical expressions for critical-band rate and critical bandwidth as a function of frequency", *Journal of the Acoustical Society of America*, **68**, 1523–1525.

[7] H. Traunmüller and F. Lacerda (1987) "Perceptual relativity in identification of two-formant vowels", *Speech Communication*, **6**, 143–157.

[8] J.D. Markel and A.H. Gray (1976) *Linear Prediction of Speech*, Springer Verlag, Berlin.

[9] R.A.W. Bladon (1982) "Arguments against formants in the auditory representation of speech", *in* R. Carlson and B. Granström (editors), *The Representation*

of Speech in the Peripheral Auditory System, pp 95–102, Elselvier Biomedical, New York.

[10] L.R. Rabiner and R.W. Schafer (1978) *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, Section 8.4, 417–421.

[11] S. Magowan (1965) "A method of plot-to-track correlation", *RRE Memo. No. 2152*, Royal Radar Establishment, Malvern, England.

Chapter 6

Multi-Processor System

Overview

This chapter describes how real-time operation of the Tutor's analysis processes can be achieved by augmenting the processing power of the PC with fast processors running concurrently. A TMS32020 board is employed to perform the data-capture and the LP analysis of the speech signal. A T800 floating-point transputer board is programmed to determine the quadratic factors of the LP polynomial and to track the resultant pole-pairs in order to determine any steady-state regions within the utterance. The transputer is also used to match the patterns obtained with a template pattern. The communication methods between the various processors are also covered.

6.1 Introduction

Chapter 4 described a minimum system for the Tutor and how a non-real-time simulation was implemented on a PC. With 8th-order LPC the simulation was capable of analysing one second of speech in approximately 25 seconds. To achieve real-time operation it is necessary to enhance the processing power of the PC.

One method of achieving this is to use the PC as a host to fast processor boards which can operate concurrently. The processes that have been described are arranged in a pipeline structure: the output of one task is passed on to another task for further processing. For this reason the system does not lend itself to operating on a true parallel arrangement of processors, but would be more easily implemented on concurrent processors arranged in a pipeline architecture.

In order to pass information through the system the pipe-head and pipe-tail must communicate with one other processor, and the remaining processors must communicate with two others. This will increase the overhead of the complete system. Therefore, the cumulative processing power of the system must be in *excess* of 25 times that of the PC AT alone. This has been achieved by augmenting the processing power of the PC with fast processors running concurrently. A TMS32020 board is employed to perform the data-capture and the LP analysis of the speech signal. A T800 floating-point transputer board is programmed to determine the quadratic factors of the LP polynomial and to track the resultant pole-pairs in order to determine any steady-state regions within the utterance. The transputer is also used to match the patterns obtained with a template pattern.

6.2 Hardware

Real-time operation of the analysis procedures can only be achieved by supplementing the processing power of the PC with fast processors running concurrently.

In addition to the tasks performed by the simulation, the system must amplify the output of a microphone, filter the subsequent speech signal to reduce the effects of aliasing, and digitally sample the speech waveform.

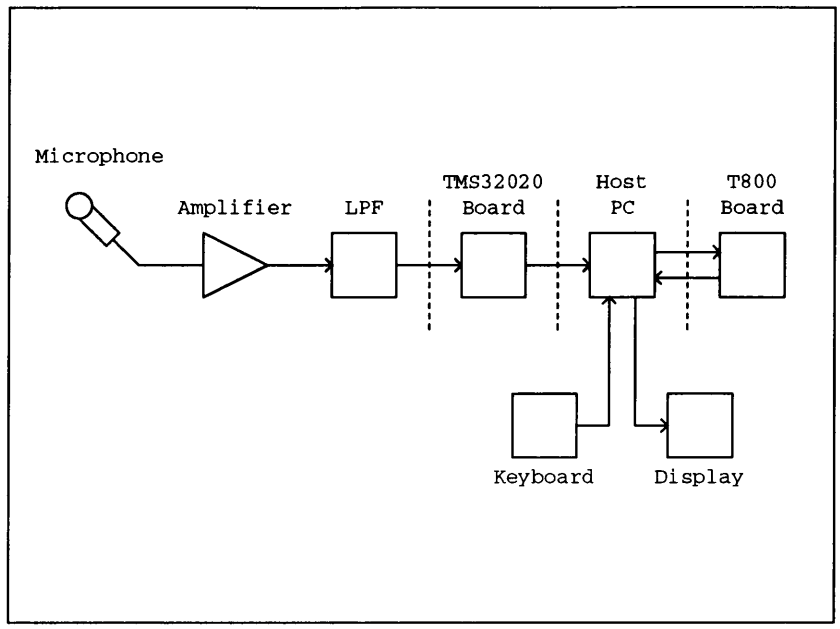

Figure 23: Hardware components of the Tutor

This section describes the choice of hardware that was used to construct a complete system, and the relationship between the hardware components is shown in Figure 23.

6.2.1 Portable Personal Computer

The host computer for the multi-processor arrangement is a *portable* PC AT Compatible. It was desired that the system be portable for two reasons. First, Tayside Health Board's Speech Therapy Department comprises many small departments situated in different hospitals and training centres: the Tutor may not always be used in the same building. Second, it is standard practice within the Speech Therapy Department to lock computer equipment away after use for security reasons. A portable system is therefore far more convenient and practical.

The portable PC has an integral monochrome liquid crystal display (LCD) which can support the screen modes available with IBM's color graphics adapter. The four colours available on the screen at any one time are represented by varying shades of grey. A sliding contrast control is supplied which caters for varying lighting conditions. However, the contrast between on and off pixels varies not only with lighting

conditions, but with viewing angle: therapists will have to be careful that the contrast is high from the patient's viewpoint and may have to put up with poorer contrast themselves. A back-lit LCD might have overcome this problem.

Secondary storage is in the form of a 5.25 inch floppy disk, and a 32 Mbyte hard disk. With a hard disk in the system, it is essential that it is *parked* prior to transporting the system. In a speech therapy application, it is likely that the system will be moved after each operation of the system, and hence a disk-parking procedure must be included in the therapy software. It would have been ideal if this was done automatically at switch-off so that the therapists do not have to remember to do something the purpose of which they may not understand. However, at the time of supplying the Tutor to the therapists for clinical trials the only disk-parking software that was immediately available offered the disk-park utility from a menu. Therefore user input was required from the therapist, and hence the process was not automatic.

To accommodate the additional hardware required, the portable PC has six expansion slots, two of which are restricted to half length by the power supply. Two of the full length slots are occupied by a disk controller and the display adapter card. This leaves two half length and two full length slots available for expansion.

The central processing unit of the system is an Intel 80286 processor which can be switched between 6, 8, and 10 MHz. The mother board is populated with 640 Kbytes of RAM.

6.2.2 Gain Control and Anti-Aliasing Filter

The only equipment external to the PC will be a microphone into which the patient will speak. The electrical signal from the microphone (in the order of μV) must be amplified to match the range of the analogue-to-digital converter. The therapist may also require that samples be taken from a recorded session with the patient. The output line voltage from a recorder is 1V pk-pk. The amplification will be switched to cater for both input levels.

To reduce the effects of aliasing, the signal must be passed through a low-pass filter (LPF) before sampling. The speech signal will be digitally sampled at 10 kHz and therefore frequencies above 5 kHz must be reduced as much as possible. This is

realised using an 8th-order (48 dB/octave fall-off) LPF with a cut-off frequency of 4 kHz.

The preferred position of the microphone, and the intensity of the speech, will vary from patient to patient. This could cause problems if a patient with a loud voice holds the microphone close. The combined amplification of the amplifier and filter is therefore set to cater for the quietest situation, and a digitally-controlled attenuator added. The speech therapy Tutor will set the attenuation to match the situation.

The amplification, anti-aliasing filter, and the gain control are all fitted on a half-length board with BNC connectors for the input and output.

6.2.3 TMS32020 DSP Board

The TMS32020 DSP board for the PC, supplied by Loughborough Sound Images Ltd, integrates Texas Instruments' TMS32020 digital signal processor with fast memory devices and data acquisition hardware. The board is full length, but due to its physical construction, requires an XT (8-bit) slot.

The TMS32020 processor is the second member of the TM320 family of digital signal processors. It is capable of performing multi-function instructions, such as a 16-bit multiply-accumulate with data move, in a single instruction cycle of 200 ns. However, for single cycle operation, it is a requirement that the data to be manipulated are located in the rather limited 544 16-bit memory locations internal to the processor.

The DSP board is mapped into 8 locations of the PC's I/O space. The starting I/O address of the block is configurable to one of 8 locations. The PC has access to the memory external to the processor via two 16-bit ports.

6.2.4 TMB08 Transputer Board

The TMB08 board from Transtech is a mother-board which can accommodate 10 T800 transputer modules (trams). Each tram has a T800, which is an INMOS transputer with a built-in hardware floating-point arithmetic unit, and can have up to 1 Mbyte of memory.

The board is full length and will operate in either an XT (8-bit) or AT (16-bit) expansion slot. The interface to the PC is compatible with the B004 module specifications from INMOS [1]. It includes a BIOS EPROM which is 32 Kbytes long and is mapped to location `0xD000`. Communications between the PC and the TMB08 can either be direct memory access or through eight 8-bit ports mapped onto the PC's I/O address. The base location for these ports is switchable between two locations.

For the initial system, this board was populated with a single T800 transputer module with 1 Mbyte of memory. If necessary, the board could be enhanced by the addition of further trams.

6.3 Distributing the Processes

For real-time operation, the analysis of the current frame must be complete before the next frame has been sampled. The desired sampling rate is 10 kHz and the frame size is 100 samples, giving a frame duration of 10 ms.

Of the tasks described in Chapter 4, the most time-critical operations are the ones that process each individual analysis frame of the utterance. Table 8 lists these processes in the order that they are executed, and shows the time taken per frame as measured on an 80287-equipped 6 MHz PC AT.

Prior to the first task shown, the amplified speech signal must be digitally sampled. This task will be carried out by the ADC on the TMS32020 board. It is therefore sensible to assign the first task, i.e. the production of the set of simultaneous linear equations, to the DSP board.

The simple distribution approach which was employed was to implement the tasks one by one on the DSP board, until it did not have the processing time left to accommodate another. The remaining tasks would then be implemented on the transputer board, which could be expanded if necessary. The PC, which is the slowest processor, would be used to facilitate data transfer between the DSP and the transputer.

The first step, therefore, is to implement the digital sampling of the speech waveform on the DSP board. The following chapter describes the implementation details of this process together with the LP analysis module.

Analysis Task	Time Per Frame (ms)
calculate covariance matrix and vector	50
solve system of linear equations	38
filter linear prediction coefficients	14
calculate poles of LP model	122
apply tracker to pole-pairs in rs-plane	22

Table 8: Timings for analysis processes required for each frame.

6.4 References

[1] INMOS (1985) *Reference Manual: Transputer*, INMOS Limited, PO Box 424, Bristol BS99 7DD, England.

Chapter 7

LP Analysis on the TMS32020

Overview

The data acquisition and LP analysis are implemented on the TMS32020 DSP board.

The later section of the LP analysis requires floating-point arithmetic. It is shown that although the IEEE standard for storing single precision numbers gives efficient use of the limited internal memory on the TMS32020, splitting the number into the relevant fields is time consuming. It is also shown that storing the numbers with all the fields stored separately requires more than the fast internal memory. A compromise is therefore necessary.

Using worst case timings, it is seen that the data acquisition and LP analysis require just under 10.5 ms, where only 10 ms of processor time is available. In practice though, it rarely takes more than 9.4 ms. So although real-time operation cannot be guaranteed, it generally will be achieved.

7.1 Introduction

As was described in the previous chapter, the signal from the microphone is amplified and filtered to reduce the effects of aliasing. The signal is then sampled at 10 kHz and divided into contiguous analysis frames each comprising 100 samples. Once a frame has been sampled it must be determined whether or not this sample constitutes a segment of an utterance. If it does it is passed through the analysis pipeline.

The first two stages in the pipeline constitute the LP analysis process which will perform 8th-order analysis on a frame size of 100 samples. LP analysis using the covariance method [1] can be split into two main sections, namely:

- generation of the covariance matrix and vector which, together with a vector of unknowns, represents a set of linear simultaneous equations.
- solution of these equations to give the LP coefficients.

The method of generating the covariance parameters is implemented first and then its execution time is determined. An estimate of the processing requirements of the solution process can then be made. If this estimate is less than the available processing time, then the solution of the equations can also be implemented on the TMS32020.

7.2 Data Acquisition

Digital sampling of the speech signal is carried out by the TMS32020 DSP board, and must be an ongoing process which is transparent to the rest of the processes. This transparency can be achieved by implementing the data acquisition process as an interrupt service routine (ISR) which is invoked at regular intervals.

The DSP board has an interval timer which can be employed to generate interrupts to the TMS32020. The interval timer is also used to initiate analogue-to-digital conversion. The resultant data is clocked into a 16-bit latch by the status signal from the ADC. A suitable ISR would therefore read the data from the ADC latch and store it in a frame buffer.

Another requirement of the ISR is that it must flag the main process to indicate that a complete frame has been sampled. The main program, at this time, must have completed working on the previous frame of data, and down-load the buffer into the main process work space before the next sample is taken. Before processing the new data, the main program must reset the flag to signal to the ISR that the buffer area is clear for the next frame.

7.2.1 TMS32020 Architecture Considerations

To ensure minimum execution time of the real-time software components that will be implemented on the TMS32020 board, it is necessary to recognise limitations of the processor and to understand how instruction times can be minimised.

The TMS32020 digital signal processor has a minimum instruction cycle time of 200 ns. This time can increase if the program memory or data memory operates with wait states. The TMS32020 DSP board, though, has been populated with static RAMs with an access time of 65 ns. This means that program and data memory access operates with zero wait states. The majority of instructions will execute in one instruction cycle, provided the data memory accessed is internal to the processor. This time can increase by one or two cycles if the data memory accessed is external. It is therefore necessary to ensure that the limited 544 bytes of internal memory are reserved for the most frequently-accessed data.

The internal memory resides in three blocks namely B0, B1, and B2. B1 and B2 always reside in the data memory map, in the ranges `0x0300–0x03FF`, and `0x0060–0x007F` respectively. B0 is a 256-word block which can be configured using the CNFP instruction to reside in program memory in the address range `0xFF00–0xFFFF` (see Table 9). After a CNFD instruction, block B0 will exist in data memory in the address range `0x0200–0x02FF` (see Table 10).

Most TMS32020 instructions are multi-function and, when preceded by the RPT or RPTK instruction (creating an *RPT pipeline*), can be repeatedly executed a specified number of times (up to a limit of 256). When operating in this mode, the majority of repeatable instructions can execute in one instruction cycle. For example, applying an autocorrelation function to 100 data points can be performed in 103

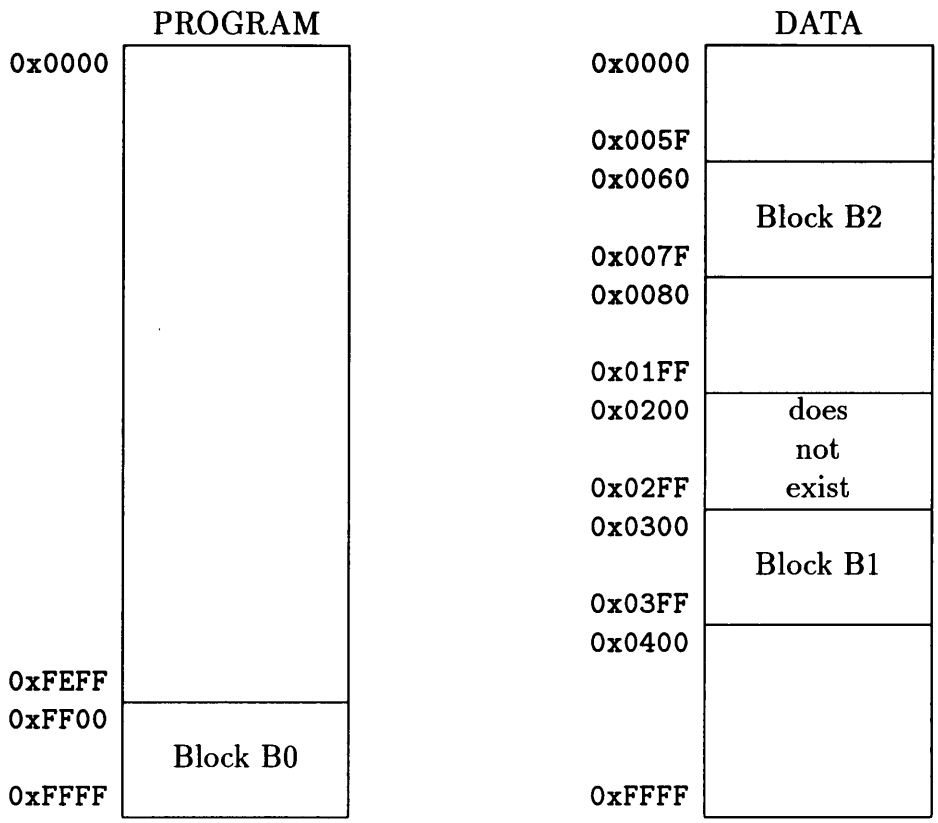

Table 9: Locations of internal memory blocks after a CNFP instruction.

instruction cycles (20.6 μs). This pipeline is protected against interrupts as no facilities are provided to save the RPT counter. It is therefore necessary to ensure that no RPT pipeline is long enough to interfere with the interrupt service routine: if the RPT pipeline takes longer than the sampling period, then a sample will be missed.

One possible limitation of the TMS32020 is the hardware stack which is used to store the program counter (PC) when subroutines are called or interrupt service routines invoked. This stack is only four words deep, and therefore only four levels of nested calls can be supported. With the data acquisition process implemented as the only ISR, then three levels of subroutine calls are left available to the main process.

When an interrupt occurs, all maskable interrupts are automatically disabled. Therefore, further interrupts must be enabled at the end of the ISR using the EINT

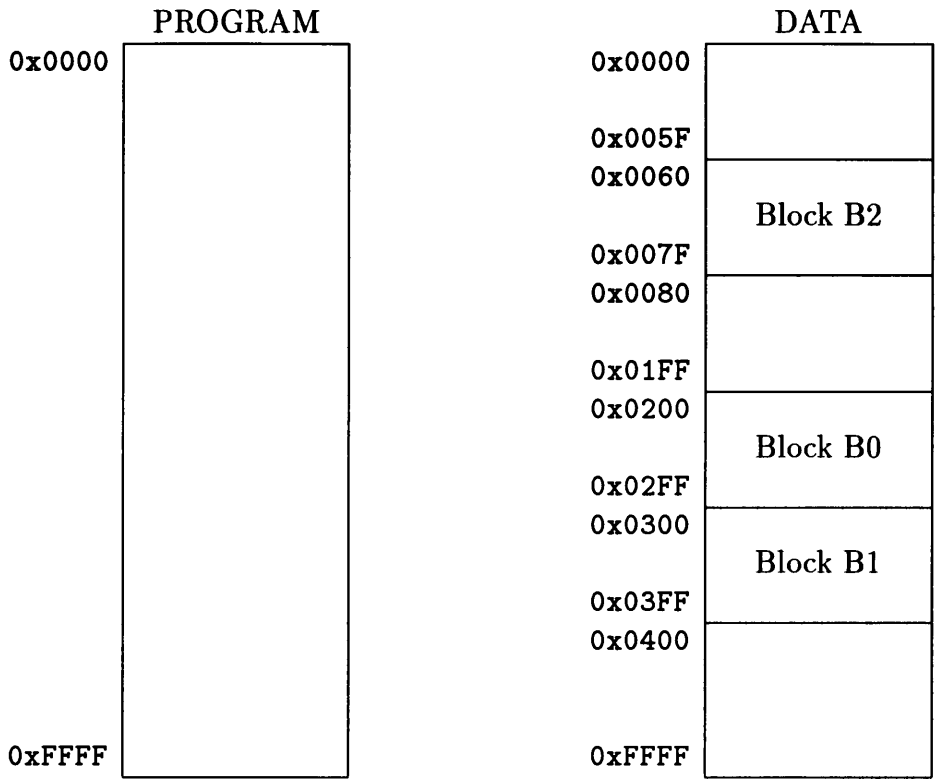

Table 10: Locations of internal memory blocks after a CNFD instruction.

instruction prior to the return from subroutine instruction RET. To prevent an interrupt being serviced before the program flow is returned to its state prior to the ISR, the TMS32020 disables interrupts for the instruction following EINT. It is very important, therefore, that the last two instructions of an ISR should be EINT and RET.

7.2.2 Implementing the ISR

The ISR must sample the speech signal at a rate of 10 kHz. The frame size is 100 samples which gives an analysis frame duration of 10 ms. When the ISR has acquired the last sample it must indicate to the main process that the current frame is complete. This can be achieved by using a communication protocol which employs a common memory location as a flag, GOTSIG.

This flag is set to `0xFFFF` by the ISR when the frame is complete and set back to 0x0000 by the main program to acknowledge the condition. To see how this protocol operates, imagine that the main program has analysed a sampled frame and is waiting for the end of the next frame to be indicated. The main program monitors GOTSIG until it changes from `0x0000` to `0xFFFF`. When this occurs, the main program sets GOTSIG back to `0x0000` and copies the frame buffer into a reserved workspace.

This protocol will only operate correctly if the main program has completed the analysis of the previous frame before the next frame has been sampled. If the main program finishes the analysis of a frame and then finds GOTSIG already set to `0xFFFF`, then it has taken longer than the permitted time to analyse the frame and a *framing error* has occurred. This framing error is registered and GOTSIG is cleared and monitored until the next frame has been sampled.

Since the ISR is invoked 100 times per frame, it is essential that its execution time is kept to a minimum. As with all interrupt service routines, all registers that are used within the ISR must be saved at the start of the routine and restored prior to returning control to the main program. As the hardware stack is limited to only four entries, the registers must be stored into reserved data memory locations. This can be facilitated using a software stack. If this stack is located in one of the internal memory blocks, then the register store instructions will execute in one cycle. Since block B2 is the smallest of the three internal memory blocks, it is most likely to be used as miscellaneous workspace. The top five locations of block B2 were therefore reserved for the software stack.

The final memory location that the ISR must access is the store for the address of the next free buffer location. With this location situated in internal memory, the total number of instructions required to service an interrupt is 33 cycles for the last sample in a frame, and 27 cycles for any other. With a frame size of 100 samples, this gives a total processor time of $(99 \times 27 + 33) = 2706$ instruction cycles, or 0.5412 ms out of a frame duration of 10 ms. This leaves an absolute maximum of 9.4588 ms to perform subsequent analysis on a sampled frame.

The sampling period is 100 μs, or 500 instruction cycles. The maximum allowable execution time for an RPT pipeline is therefore $(500 - 33) = 467$ instruction cycles

(see Section 7.2.1).

7.3 Utterance Detection

The data-acquisition process has been implemented on the DSP board, and utilises approximately 5% of available processor time. Before the first stages in the process pipeline can be implemented, it must be decided whether the sampled frame constitutes part of the utterance or is simply background noise.

The algorithm for detecting the start and end points of the utterance was discussed in Chapter 4. It required the computation of the short-time signal energy for the sampled frame using a special case of the autocorrelation function.

This energy measure can also be used to detect conditions where the results of subsequent analysis may be unreliable due to arithmetic overflow. Such conditions will occur if the speech signal is of sufficient amplitude to saturate the accumulator during repeated multiply-accumulate operations.

Computing the short-time signal energy is a sum-of-squares process. The SQRA instruction of the TMS32020 is ideally suited to this type of operation. Due to the pipeline architecture of the component modules in the central arithmetic/logic unit in the TMS32020, it can perform an accumulate operation simultaneously with a multiply operation. Note, though, that the accumulate operation uses the accumulator and the result of the *previous* product as its operands, the result being stored in the accumulator. Therefore, both the accumulator and the product register must be cleared before performing repeated SQRAs.

The current auxiliary register is used to point to the operand of SQRA. When in repeat mode the auxiliary register can be made to increment automatically, and each individual SQRA operation can be performed in a single instruction cycle. However, for SQRA to operate as a single cycle instruction, the data must reside in internal memory, otherwise it takes two instruction cycles.

The short-time signal energy is calculated over the full 100 sample frame. The SQRA instruction can be applied to 100 data points in 100 cycles (or 20 μs). This is far shorter than the sampling time period of 100 μs, and therefore will not interfere

with the interrupt service routine (see Section 7.2.1). Including the initialisation of the registers, and the storage of the result, the short-time signal energy of 100 data samples can be calculated in 107 cycles, or 21.4 μs.

The remainder of the algorithm uses this energy measure to establish whether or not the frame is part of the utterance. The longest path through this takes 345 cycles, or 69 μs, giving a total time for the algorithm of 90.4 μs. The total burden on the processor, including the data acquisition, is therefore 541.2 + 90.4 = 631.6 μs. This leaves 9.3684 ms of processor time to perform subsequent analysis.

7.4 Covariance Matrix and Vector Generation

The covariance matrix and vector represents a set of linear simultaneous equations that must be solved to give the LP coefficients. These covariance parameters are generated using the modified autocorrelation function (see Chapter 4) which is a sum-of-products operation.

The simultaneous equations are represented by the covariance matrix, Φ, the covariance vector, ϕ, and the vector of the unknown LP coefficients, α. ϕ and Φ are constructed using the modified autocorrelation function [1] which is a sum-of-products operation, and $\phi(i) = \Phi(i,0) = \Phi(0,i)$.

The TMS32020 instruction specifically for such operations is MAC (multiply-accumulate). As with the SQRA instruction (see Section 7.3), it will accumulate the *previous* product and perform signed 16-bit multiplication simultaneously. The two registers which point to the multiplicative operands are the program counter (PC) and the current auxiliary register (AR). When in repeat mode the PC and the AR can be made to increment automatically, and each single MAC operation can be performed in a single instruction cycle.

For the MAC instruction to operate in such a manner, one set of data must reside in the internal memory block B0 configured as program memory (`0xFF00–0xFFFF`) and the other set of data must reside in internal memory block B1 which is always data memory (`0x0300–0x03FF`). Table 11 shows the program and data memory usage for the calculation of the covariance elements.

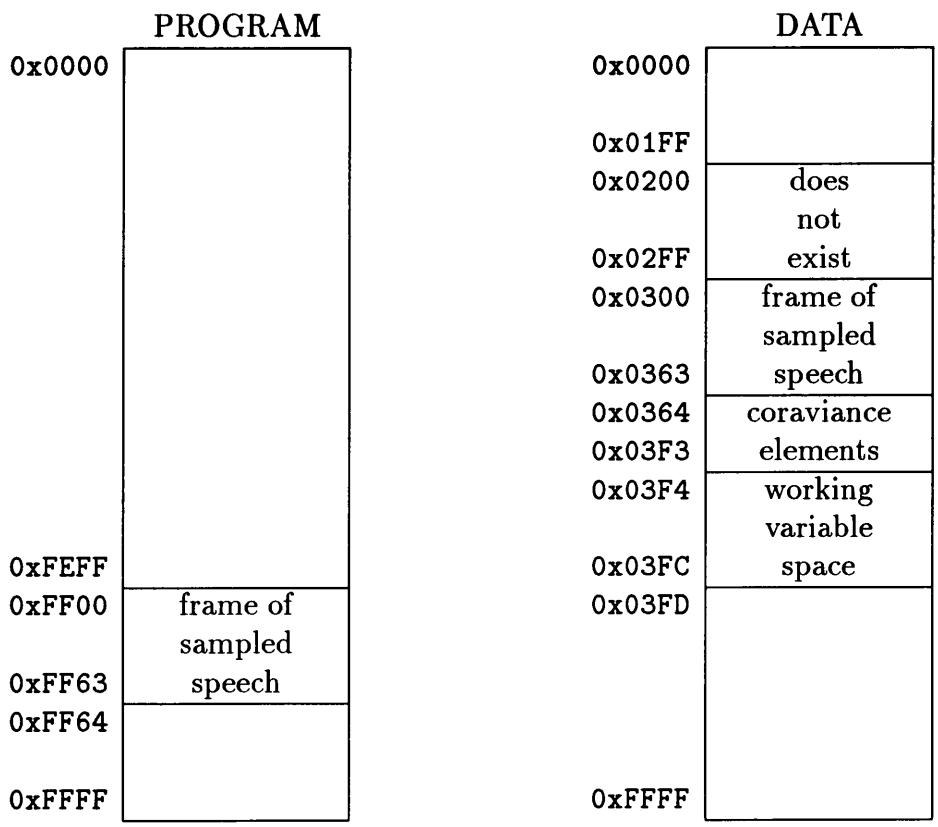

Table 11: Program and data memory maps during construction of covariance matrix and vector.

$\Phi_{1,0} \leftrightarrow$ 0x0364	$\Phi_{1,1} \leftrightarrow$ 0x0366			
$\Phi_{2,0} \leftrightarrow$ 0x0376	$\Phi_{2,1} \leftrightarrow$ 0x0378	$\Phi_{2,2} \leftrightarrow$ 0x037A		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
$\Phi_{8,0} \leftrightarrow$ 0x03E2	$\Phi_{8,1} \leftrightarrow$ 0x03E4	$\Phi_{8,2} \leftrightarrow$ 0x03E6	$\cdots$	$\Phi_{8,8} \leftrightarrow$ 0x03F2

Table 12: Storage of covariance elements as 16-bit signed integers.

The two sets of data must reside one in each memory map. This is due to the Harvard architecture of the TMS32020: the program counter and the data pointer can operate independently giving the fast instruction time. Therefore, to use the MAC instruction to perform the autocorrelation function, the sampled speech data must be copied into B1 and an identical set of data copied into B0.

To enable the process to operate at maximum speed, the elements of Φ and ϕ should also be stored in internal memory. Block B0 must be configured as program memory during the autocorrelation computation, so block B1 is used. The first 100 words of data block B1, however, are reserved for the copy of the speech signal, leaving 156 words free.

The speech is sampled with a resolution of 16 bits. The autocorrelation function will therefore produce a 32-bit result in the accumulator, so each element in Φ and ϕ can be stored as two words. ϕ comprises 8 elements, and Φ comprises 64. Therefore ϕ and Φ can be stored in 144 of the remaining 156 words of B1.

Φ is symmetrical about the leading diagonal, so only 36 of its elements need be calculated, giving a total of 44 covariance elements. It is convenient to leave these elements stored in a (8×9) array. As can be seen in Table 12, the address of $\Phi_{i,j}$, Addr($\Phi_{i,j}$), is given by:

$$\text{Addr}(\Phi_{i,j}) = \texttt{0x0364} + [(i-1) \times \texttt{0x12}] + [j \times \texttt{0x2}]$$

From the modified autocorrelation function, it can be shown that (see also Appendix B):

$$\Phi_{i-1,j-1} = \Phi_{i,j} - s(-i)s(-j) + s(N-i)s(N-j) \tag{10}$$

The construction of ϕ and Φ can therefore be split into two parts. Firstly, $\Phi_{8,0}$ to $\Phi_{8,8}$ can be calculated using the modified autocorrelation function. Secondly, the remaining elements can be calculated a row at a time using the relationship shown in equation 10.

Using this method, the first nine covariance elements can be calculated in 206.2 μs, and the remaining 35 elements can be calculated in 222.2 μs. The data acquisition and utterance detection leaves 9368.4 μs. Therefore, after the production of the set of linear simultaneous equations, there is $9368.4-(206.2+222.2) = 8940$ μs of processor time left to solve the equations.

7.5 Obtaining the LP Coefficients

Due to the nature of the covariance matrix and vector, one of the most efficient methods for solving the set of linear simultaneous equations is Cholesky's method of matrix decomposition and back substitution [1].

The simultaneous equations can be expressed in matrix form as:

$$\Phi\alpha = \phi$$

Solution of the linear simultaneous equations using Cholesky's method is a three-stage process. First, the matrix of covariance parameters, Φ, is decomposed into a triangular matrix, V, and a diagonal matrix, D, such that:

$$\Phi\alpha = VDV^t\alpha = \phi$$

Second, due to the triangular form of V, a vector Y can be calculated such that:

$$VY = \phi$$

where

$$Y = DV^t\alpha$$

Finaly, α is calculated from the above equation.

Operation	Execution time (μs)
addition	15.4
multiplication	7.8
division	22.8

Table 13: Execution times of Crowell's routines.

7.5.1 Real-Time Analysis and FP Arithmetic

Floating-point arithmetic facilitates the implementation of Cholesky's method because there is a large dynamic range of intermediate results during the calculations. However, the implementation of the floating-point routines themselves is not a trivial task.

Crowell [2] describes floating-point arithmetic routines for the TMS32020. The precision of the routines conform to that of the IEEE standard for single-precision numbers [3], but the internal representation differs to ease the access to the component parts of the number, i.e. the sign-bit, the exponent, and the mantissa.

The given execution times for these routines are shown in Table 13. The time for subtraction is not given by Crowell, but the only computation required over that of the addition routine is the negation of the sign-bit of the second operand.

To estimate the feasibility of using Crowell's routines in the real-time application, a program, `LPCFLOPS.PAS`, was written which determines the number and nature of the arithmetic operations required to determine the LP coefficients from the covariance matrix and vector. The results for an 8th-order model are given in Table 14, together with the total required processor time for that type of operation.

It can be seen that the total processor time required to perform the necessary arithmetic operations is 5370.8 μs. This leaves $(8940 - 5370.8) = 3569.2$ μs to convert the covariance matrix and vector into floating-point format, and perform data management, i.e. passing the operands to the arithmetic routines, and storing the result.

However, when applying test data to Crowell's routines, it was apparent that they

Operation	Number required	Time required (μs)
addition	140	2156.0
subtraction	42	646.8
multiplication	224	1747.2
division	36	820.8

Table 14: Number of required arithmetic operations, and the processor time required.

were flawed. For example, in his division routine he makes the sign of the result equal to the sign of the first operand if the sign of both operands are the same, otherwise the sign of the result is set to -1. This is obviously incorrect: if both a and b are negative then a/b is positive, not negative as Crowell would have us believe.

The above error may be obvious and simple to rectify, however some other fundamental errors were not so easy to correct. For this reason, the algorithms were adhered to (with a few additions to cater for cases not covered), but the implementation of the algorithms was carried out without reference to Crowell's work.

7.5.2 IEEE Single-Precision Arithmetic

The IEEE single-precision format is a 32-bit format which comprises a sign-bit, s, an offset eight-bit exponent field, e, and a 23-bit fraction part, f[3]. A floating-point number, x, is determined from s, e and f, using the following equation

$$x = (-1)^s \times 2^{(e-127)} \times 1.f$$

The three fields are combined to give a 32-bit number, as shown in Figure 24. The advantage of this 32-bit format is that it is compact: each floating-point number requires only two words for storage. This is a benefit when considering the limited amount of fast internal memory on the TMS32020.

However, before performing a floating-point operation, the fields of the two operands must be split into usable items such as a sign word, s, an exponent word, e, and two mantissa words, f_{lo} and f_{hi}. This takes 4.6 μs for each operand. Assuming

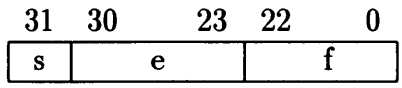

Figure 24: IEEE single-precision format.

a similar time to re-combine the components of the result, the decomposition and re-combining for the 442 arithmetic operations requires approximately 6000 μs of processor time.

This leaves under 3000 μs to perform the arithmetic and the data management. On the assumption that the correctly-implemented floating-point routines will have an execution time similar to that of Crowell's, it is not possible to use this format in the real-time system.

The most obvious solution is to store the floating-point numbers in their component parts. This requires 4 words for each number. The solution of the simultaneous equations requires two $p \times p$ triangular matrices, and four $p \times 1$ vectors (see Section 7.5). However, there is not enough internal memory to keep the indexing simple by arranging triangular matrices as two-dimensional arrays.

A compromise, though, can be reached. The combination of f_{hi} and s reduces the number of words required to three, as shown in Figure 25. The 1 between s and f_{hi} is a by-product of the normalisation process which ensures that the mantissa is greater than or equal to 1.0 and less than 2.0.

The precision of this format is identical to that of the IEEE single-precision standard. However, since only s and f_{hi} are combined, this alternative representation reduces the time taken to obtain the component parts of the number. The worst case execution times of the floating-point routines which utilise this format are shown in Table 15, and the total processor time required for each type of iteration is shown in Table 16.

The timings shown are for the worst case. The maximum total processor time required for the arithmetic is therefore 6405.2 μs. The time left for normalising the covariance matrix and vector and to manage the data is 2534.8 μs.

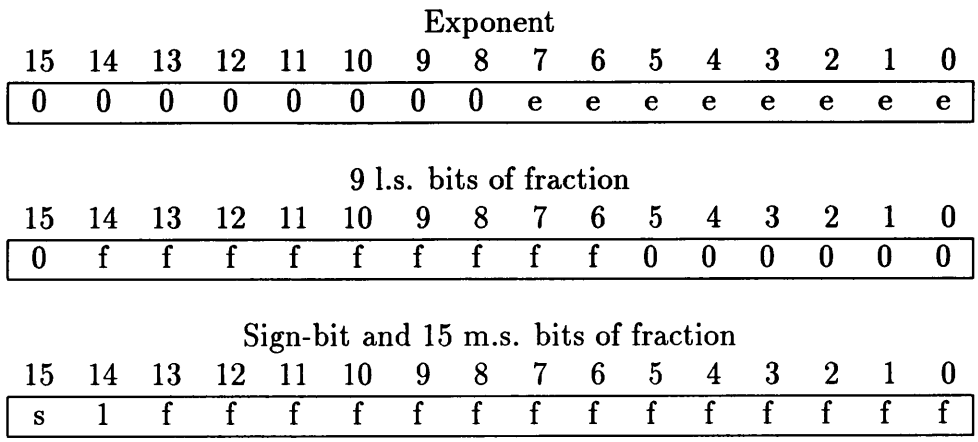

Figure 25: Alternative floating-point representation.

Operation	Execution time (μs)
addition	20.0
subtraction	21.0
multiplication	9.2
division	18.4

Table 15: Execution times of the new floating-point routines.

Operation	Number required	Time required (μs)
addition	140	2800.0
subtraction	42	882.0
multiplication	224	2060.8
division	36	662.4

Table 16: Number of required arithmetic operations, and the processor time required.

7.5.3 Covariance Matrix and Vector Conversion

The covariance matrix and vector which represent a system of linear simultaneous equations are initially calculated using two's complement integer arithmetic. Before the equations can be solved to give the LP coefficients, these values must be converted to floating-point numbers.

For the floating-point representation, the fractional part (which incorporates the sign-bit) is normalised i.e. the most significant bit, the sign-bit, and the second most significant bit are not equal. Zero is the only exception, which is considered normalised.

The first step in producing the floating-point covariance elements is to normalise the initial fixed-point results. The TMS32020 provides an instruction NORM which facilitates this process. A NORM instruction will shift the accumulator left if the accumulator is not normalised and increment the current auxiliary register, otherwise it will leave the accumulator unaltered.

In this instance, negative numbers are negated and the sign-bit altered accordingly after normalisation. The maximum number of repeated NORMs required to normalise fully the smallest positive number, `0x00001`, is 30 (producing `0x4000`). Therefore the current auxiliary register, AR_n, can be set to zero, and the NORM instruction repeated 30 times, the auxiliary register recording the number of normalisation cycles that were effective. AR_n is therefore related to the negative of the exponent, as the normal method of producing a floating-point number is repeatedly to *divide* the number by the base while incrementing the exponent. The exponent can be calculated as $(30 - AR_n)$.

After normalisation, the most significant word of the accumulator is equal to f_{hi}, and f_{lo} is obtained by shifting the low word of the accumulator right one place, and masking off the most significant and six least significant bits.

To maximise the speed at which the integer values can be converted to their floating-point representation, the results must be stored in internal memory. Both copies of the frame of sampled speech that were required for the generation of the covariance parameters are no longer needed. This frees the first 100 words of block B2 and all of block B1. Block B1 can now be configured as data memory. Each

$\Phi_{1,0} \leftrightarrow$ 0x0200	$\Phi_{1,1} \leftrightarrow$ 0x0203				
$\Phi_{2,0} \leftrightarrow$ 0x0218	$\Phi_{2,1} \leftrightarrow$ 0x021B	$\Phi_{2,2} \leftrightarrow$ 0x021E			
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
$\Phi_{7,0} \leftrightarrow$ 0x0290	$\Phi_{7,1} \leftrightarrow$ 0x0293	$\Phi_{7,2} \leftrightarrow$ 0x0296	$\cdots$	$\Phi_{7,7} \leftrightarrow$ 0x02A5	
$\Phi_{8,0} \leftrightarrow$ 0x02A8	$\Phi_{8,1} \leftrightarrow$ 0x02AB	$\Phi_{8,2} \leftrightarrow$ 0x02AE	$\cdots$	$\Phi_{8,7} \leftrightarrow$ 0x02BD	$\Phi_{8,8} \leftrightarrow$ 0x02C0

Table 17: Storage of covariance elements as floating-point numbers.

floating-point covariance element requires 3 words, and to simplify the access to the elements, they could be stored as a (9×8) array.

However, a more efficient use of the valuable internal memory is shown in Table 17. With each element requiring 3 words, and each row taking 0x18 words, it can be seen that the covariance matrix and vector are stored in an (8×8) array. This storage technique utilises the fact that the covariance matrix exhibits diagonal symmetry, and so not all of its elements need be stored. The eighth row is the only one that requires 9 elements to be stored. Therefore the final element, $\Phi_{8,8}$, is actually outside this array but is addressed in the same way as all the others, using the following equation:

$$\text{Addr}(\Phi_{i,j}) = \texttt{0x0200} + [(i-1) \times \texttt{0x18}] + [j \times \texttt{0x3}]$$

When the conversion is complete, all of block B2 is available for data storage.

The time taken to convert a 32-bit number into a floating-point number is 10.6 μs for positive definite numbers, 11.4 μs for negative definite numbers, and 3.4 μs for zero.

The array of covariance elements consists of 9 rows, with a total of 44 unique entries. Initialising constants for the operation takes 3.4 μs. Initialising the pointers for each row takes 3.8 μs. Assuming the worst case, the time required to convert the covariance elements to floating-point numbers is $(3.4 + 9 \times 3.8 + 44 \times 11.4) = 539.2$ μs. This leaves $(2534.8 - 539.2) = 1995.6$ μs to manage the data transfers between floating-point operations which constitute the solution of the linear simultaneous equations.

$D_1 \leftrightarrow$ 0x0300	$D_2 \leftrightarrow$ 0x0303	$\cdots$	$D_8 \leftrightarrow$ 0x0315
$Y_1 \leftrightarrow$ 0x0318	$Y_2 \leftrightarrow$ 0x031B	$\cdots$	$Y_8 \leftrightarrow$ 0x032D
$\alpha_1 \leftrightarrow$ 0x0330	$\alpha_2 \leftrightarrow$ 0x0333	$\cdots$	$\alpha_8 \leftrightarrow$ 0x0345
$V_{1,1} \leftrightarrow$ 0x0348			
$V_{2,1} \leftrightarrow$ 0x035D	$V_{2,2} \leftrightarrow$ 0x0360		
$\vdots$	$\vdots$	$\ddots$	
$V_{8,1} \leftrightarrow$ 0x03DB	$V_{8,2} \leftrightarrow$ 0x03DE	$\cdots$	$V_{8,8} \leftrightarrow$ 0x03F0

Table 18: Storage of D, Y, α, and V.

7.5.4 Implementing Cholesky's Method

The first consideration in implementing Cholesky's method is the data memory requirement. Solution using Cholesky's method requires the covariance matrix and vector, Φ and ϕ, a triangular matrix, V, a diagonal matrix D, an intermediate vector Y, and the output vector, α. D can be treated as a vector for storage purposes since the only non-zero elements are those in the leading diagonal.

Φ and ϕ are stored in data block B2 as shown in Table 17. The internal memory block B1 is free for the storage of the remaining matrix and vectors. D, Y, and α, are stored as linear one-dimensional arrays, each requiring 24 words of memory. Reserving the first 72 locations of B1 for these vectors leaves 184 words for V. To simplify the addressing of the elements of V it is convenient to store it as an (8×8) array. This, though, would require 192 words. However, using the same technique as used for the storage of ϕ and Φ, this can be reduced to 171 words. The memory usage for D, Y, α, and V is shown in Table 18. Each element of V is addressed using the following equation:

$$\text{Addr}(V_{i,j}) = \texttt{0x0348} + (i-1) \times \texttt{0x15} + j \times \texttt{0x3}$$

Cholesky decomposition involves the calculation of D and V. The elements of V are determined using:

$$V_{ji} = \frac{\Phi_{ji} - \sum_{k=1}^{i-1} V_{jk} D_k V_{ik}}{D_i} \qquad 2 \le i \le j-1 \tag{11}$$

Implementation of	Execution time (instruction cycles)
equation 11	$63 + i \times 64$
equation 12	$42 + i \times 51$

Table 19: Execution times of equations 11 and 12.

with:

$$V_{j1} = \frac{\Phi_{j1}}{D_1}$$

and the elements of the diagonal matrix calculated using:

$$D_i = \Phi_{ii} - \sum_{k=1}^{i-1} V_{ik}^2 D_k \qquad i \geq 2 \tag{12}$$

with:

$$D_1 = \Phi_{11}$$

Equations 11 and 12 were implemented as subroutines, which both have execution times dependent on i, as shown in Table 19.

Bearing in mind that all calls to the floating-point arithmetic routines have previously been calculated (for the worst case) in Section 7.5.2, the time taken to manage the data in the decomposition process must be calculated.

Copying $\Phi_{1,1}$ to D_1 and the common divisor for V_{j1} takes 16 instruction cycles. Calculating the first row of V takes a further 188 instruction cycles. Calculating by hand the remaining execution time for Cholesky decomposition is complex, as it comprises a nested loop: for each i, D_i is calculated, and j is varied to calculate V_{ji}. However, a short Pascal program was written, `DECOMPCY.PAS`, which emulates the nested loop and accumulates the number of instruction cycles required. The result is that the Cholesky decomposition process takes 8727 instruction cycles, or 1745.4 μs.

The total free processor time prior to this stage in the analysis is 1995.6 μs (see Section 7.5.3). This leaves a mere 250.2 μs to apply the Cholesky substitution process.

Cholesky substitution can be split into two sections. First, the elements of the

intermediate vector, Y, are calculated using the equation:

$$Y_j = \phi_j - \sum_{i=1}^{j-1} V_{ji} Y_i \qquad 2 \leq j \leq 8 \tag{13}$$

with:

$$Y_1 = \phi_1 \tag{14}$$

Second, the LP coefficients, α, are determined using the equation:

$$\alpha_i = \frac{Y_i}{D_i} - \sum_{j=i+1}^{8} V_{ji} \alpha_i \qquad 7 \geq i \geq 1 \tag{15}$$

with:

$$\alpha_8 = \frac{Y_8}{D_8} \tag{16}$$

The implementation of equation 14 takes 11 clock cycles. Equation 13 is implemented as nested loops. The outer loop, with index j, is executed 7 times. Initialising the loop variable takes 2 instruction cycles, and the loop body takes 49 cycles, including the initialisation of the inner loop variable, i, but excluding the execution of the inner loop body.

The inner loop body is executed once for the first execution of the outer loop, twice for the second and so on up to seven times for the last outer loop. This gives a total of 28 executions of the inner loop, each taking 51 instruction cycles. The calculation of the intermediate vector, Y, therefore takes a total of $(11+2+7\times 49+28\times 51) = 1784$ instruction cycles.

The first LP coefficient, α_8, is calculated using equation 16 in 26 instruction cycles. Nested loops are also required for the implementation of equation 15. Initialising the outer loop variable, i, takes 2 cycles. The outer loop body is executed seven times, with an execution time of 64 cycles.

The inner loop is executed once for the first iteration of the outer loop, twice for the second, and so on up to 7 executions for the last one. This again gives 28 executions of the inner loop. With the inner loop body taking 52 cycles, the total time taken for the calculation of the LP coefficients, α, is $(26+2+7\times 64+28\times 52) = 1932$ instruction cycles.

The Cholesky substitution process therefore takes 3716 instruction cycles, or 743.2 μs. The total time taken to sample one frame of speech and calculate the LP coefficients is therefore 10.493 ms out of a maximum permissible frame duration of 10 ms.

Thus, the LP analysis on the TMS32020, cannot be guaranteed to operate in real-time. Recollect, though, that the timings for the floating-point routines were stated as worst-case instances. For example, if the addition routine exits via 4 out of its possible 5 end points, it will operate in under 12 μs, as opposed to the stated 20 μs.

7.6 Timing and Verifying the LP Analysis

It is difficult to predict how the LP analysis will perform when speech data is applied. However, the TMS32020 digital signal processor has an on chip memory-mapped 16-bit timer. This timer decrements at a rate of once every four instruction cycles, and can therefore be utilised to time programs to within four instruction cycles, or 0.8 μs.

In order to determine the execution time of the LP analysis on one frame of speech data, it is convenient to have the software running as a one-shot system, rather than continually as in the Tutor. It is then possible to determine the accuracy of the LP analysis on the DSP by applying a known set of data and comparing the results obtained with those from the simulation routines.

Several monosyllabic utterances were sampled and stored on hard disk, so that they could be passed to the LP analysis one frame at a time. Although the data acquisition interrupt service routine is not then required, it is kept active to make the timing as close as possible to that for the Tutor. To achieve this, the ISR was altered so that the data read from the analogue-to-digital converter were not stored. The timing of the routine, however, is left unaltered. The utterance detection routine was similarly altered so that each frame of data is accepted but the timing preserved.

The short-time signal energy was calculated for each frame. If this measure showed a maximum value of `0x7FFF` then it was likely that the accumulator saturated during the computation. In such a situation, it is also possible for the accumulator to have saturated during the calculation of the covariance parameters. Therefore, if an

Minimum	Average	Maximum
9206.4	9347.1	9448.8

Table 20: Maximum, minimum, and average execution times, in μs, over 548 frames of speech data.

energy overflow was detected the timing and analysis results were discarded from the following statistical analysis.

For 548 frames of speech analysed, the maximum, minimum, and average execution times are shown in Table 20. The maximum recorded time is well below the maximum permissable time of 10000 μs.

This means that although real-time operation cannot be *guaranteed*, it is highly unlikely that the maximum permissable time will be exceeded resulting in framing errors.

To determine the accuracy of the DSP LP analysis module, the resultant LP coefficients were compared with those for the simulation routines. The order of the error was estimated by taking the logarithm, to base ten, of the absolute value of the error, and taking the nearest integer result, i.e. the order of the error can be calculated in Pascal as

```
Order := Trunc(-Ln(Error)/Ln(10)+0.5);
```

This means that if the order of the error, e, is equal to the integer n then:

$$\sqrt{10}^{n-1} \leq e \leq \sqrt{10}^{n}$$

The numbers of occurrences of errors of orders in the range 2 to 10 inclusive are shown in Table 21. There were no occurrences of orders outside this range. It can be seen that for less than 0.5% of the time, the error was within the maximum range of:

$$3.16 \times 10^{-3} \leq e \leq 3.16 \times 10^{-4}$$

Order of Error	2	3	4	5	6	7	8	9	10
No. of occurrences	0	19	404	2727	1078	142	13	1	0

Table 21: Order of errors in the LP analysis module implemented on the TMS32020.

Register Name	Port Address	Function
DSPDATA	0x292	16-bit data
CONTROL	0x294	8-bit control register
DSPADDR	0x296	16-bit address

Table 22: PC port addresses of DSPDATA, DSPADDR, and CONTROL.

7.7 The DSP-to-PC Interface

The communication channel between the DSP board is bi-directional but is essentially under the control of the host PC: the PC can read or write to the DSP's memory but no facilities are supported which allow the DSP access to the host's memory.

The PC can access all program and data memory that does not constitute part of the DSP's 544 words of internal memory. When the PC accesses the available memory, the DSP operation is suspended for one wait state (200 ns).

The PC accesses the DSP board's memory through a pair of 16-bit registers which are mapped into the PC's I/O space. The port addresses of these registers can be altered using a bank of links on the DSP board, but for this application they are left at the default addresses shown in Table 22.

DSPDATA holds the data read from, or to be written to, DSP memory. The write-only register DSPADDR controls the initial value of an up/down counter which, in turn, determines the address of the memory location to be accessed. This reduces the overall access time with respect to the PC if consecutive words of memory are to be accessed.

The selection of program or data memory and the mode of the up/down counter is controlled via an 8-bit write-only control register, CONTROL, the address of which is shown in Table 22. This register is also used to reset or hold the TMS32020. The

Bit 76543210	Function
01XXXXXX	reset DSP
10XXXXXX	hold DSP
11XXXXXX	run
XX0XXXXX	up count
XX1XXXXX	down count
XXXX0XXX	data memory
XXXX1XXX	program memory

Table 23: Operation of the control register.

required states of the relevant bits of this register are shown in Table 23.

7.8 DSP-to-PC Communications Protocol

Once the LP coefficients for a particular frame have been determined, they must be transferred to the transputer board for further processing. Included in this subsequent analysis is the root-finding process (see Chapter 4) which is iterative in nature; therefore, there can be no guarantee that the process will be completed within the duration of the analysis frame.

For this reason, it is necessary to insure that any delay in the transputer processing in no way affects the LP analysis. To cater for such a situation the results of the LP analysis are stored in contiguous blocks of memory on the DSP board. This means that the DSP board is not dependent on the state of either the PC or the transputer. The PC on the other hand is wholly dependent on the state of the DSP. If the PC has completed dealing with the previous frame then it must wait until the DSP board has completed analysing the current frame before proceeding.

Therefore, it is a requirement that each block of data contains a flag which indicates whether or not the block is valid data: once the PC has dealt with the previous frame, it reads continually this flag until it indicates that the data is valid.

This flag, which will reside in external memory on the DSP card, can be read

repeatedly until it indicates that the results for that frame are valid. However, since reading the DSP memory from the PC inserts one wait state for each memory access, the communications will effectively reduce the available processing time per frame. It is therefore necessary to estimate this reduction.

A simple test program was written, `TIMEPORT.PAS`, which times 10^6 accesses to the same DSP memory location. The program took 58.17 seconds to run. Removing the memory-access routine showed that 10.22 seconds of this time could be attributed to the loop structure alone.

One million memory accesses, therefore, takes 47.95 seconds. So, in one frame of 10 ms, there can be a maximum of 209 memory accesses. This reduces the processing time per frame by 209 wait states, or 41.8 μs. Allowing 9.5 ms for the analysis of each frame of speech data, this additional burden reduces the safety margin of 0.5 ms by approximately 10%, and is therefore acceptable.

7.9 Conclusions

For real-time operation with a sampling rate of 10 kHz and a frame size of 100 samples, the data acquisition and LP analysis implemented on the TMS3202 must execute in under 10 ms for each frame.

It can be guaranteed that these two processes will operate in under 10.5 ms; this does not give a real-time system. However, using a large number of frames of sampled speech, it is likely that the data acquisition, LP analysis, and PC–DSP communications require under 9.5 ms of processor time. Therefore, although real-time operation cannot be guaranteed, in practice it will be achieved.

7.10 References

[1] L.R. Rabiner and R.W. Schafer (1978) *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, Section 8.4, 417–421.

[2] C. Crowell (1986) "Floating point arithmetic with the TMS32020", *Digital Signal Processing Applications*, Texas Instruments Inc, France.

[3] Draft 8.0 of IEEE Task P754 (1981) "A proposed standard for binary floating-point arithmetic", *IEEE Computer*, March 1981, 51–62.

Chapter 8

Real-Time Analysis on the Transputer

Overview

This chapter describes how the transputer (T800) was employed to smooth the LP coefficients, determine the quadratic factors of the modified LP polynomial, and track the resultant pole-pairs in real-time. The recorded tracks are then analysed to determine any steady-state regions within an utterance. The PC-to-T800 communications protocol is also covered by this chapter.

8.1 Introduction

Chapter 7 showed that real-time production of the LP coefficients can be achieved using the TMS32020 DSP board. The only channel open for transferring these LP coefficients to the transputer board is via the PC. As can be seen from Section 6.3, the PC does not run fast enough to filter the coefficients after reading them from the DSP, prior to transferring them to the transputer board.

The responsibility of the transputer board is therefore to read the coefficients from the PC and filter the coefficients to ease the tracking process. The poles of the LP model are then found and tracked to identify any steady-state regions in the utterance. Finally, the transputer matches the results with a template to estimate the quality of the utterance.

The filtering, root-finding and tracking must be done in, or very close to real-time, to minimise the delay between the end of the utterance and the displaying of the results. The identification of the steady-state regions and the template match, however, are done at the end of the utterance. Since the template-matching process is identical to the comparison made between successive frames in the tracking process, there is no need to investigate it explicitly.

This chapter, therefore, is concerned with the real-time aspects of the analysis procedure. Various communications methods will be investigated, and awarded merits according to the speed at which they can pass information from the DSP to the transputer. All timings quoted are for a 6 MHz IBM PC AT.

The first stage in determining whether or not real-time operation is achievable using the available hardware setup is to estimate the transputer processing time required to perform the analysis tasks.

8.2 Timing the Tasks on the Transputer

The filtering, factorising, and tracking procedures were taken from the non-real-time simulation and translated into C to run on the transputer (the particular implementation of C chosen for all transputer application programs is 3L's Parallel C). These

Process	Cumulative Time (sec)	Individual Time (sec)	Time per Frame (ms)
simulate LPC	4	—	—
filtering	67	63	0.63
factorising	224	157	1.57
tracking	315	91	0.91

Table 24: Cumulative times of processes on the transputer.

procedures were first incorporated into a test program `PROCTEST.C` which supplied the procedures with a known data set. The simulation procedures were incorporated into an equivalent Pascal program `PROCTEST.PAS`. The C routines were validated by comparing the output from both these programs which were found to be identical.

`PROCTEST.C` was then modified to produce `TIMEPROC.C` which times 10^5 repetitions of the analysis pipeline. The program first times the simulation of the reading of the LP coefficients (they are actually read from a cyclic buffer of 50 sets of eight coefficients). It then adds the first analysis process (the filtering of the LP coefficients) and displays the cumulative time, i.e. the time taken for 10^5 repetitions of the LP simulation *and* the filtering. The program then systematically adds in one further analysis process and displays the cumulative time. Table 24 shows the result. The first column shows the cumulative time up to and including each stage in the analysis pipeline for 10^5 simulated frames. The second column shows the individual process time for the 10^5 frame simulation. The third column shows the individual time of each process for one frame.

As can be seen, the total time taken per frame to filter the LP coefficients, factorise the LP polynomial to obtain the pole-pairs, and track these pole-pairs is just 3.11 ms per frame. Of the tasks that are to be performed for each analysis frame, all that remains to be implemented is a method of reading the LP coefficients from the DSP memory into the transputer memory.

8.3 T800 Board Communication Channel

T800 application programs are loaded into the T800 board using AFSERVER which runs on the host PC. AFSERVER is supplied as standard with the Parallel C language and is an MS-DOS executable file which is produced by INMOS, the developers of the transputer. It is used to load and run a transputer program, and normally stays resident in the host's memory while the transputer application program is running, allowing the transputer to communicate through it to the host's peripherals.

As AFSERVER is active for the duration of the transputer application program, it is difficult to use the PC to do any of the processing in the pipeline structure. It is therefore necessary for the T800 board to communicate with the DSP board in order to get the LP coefficients for further processing.

8.3.1 Communicating through AFSERVER

The T800 application was written in 3L's Parallel C language. Its input-output functions conform to the Kernighan and Ritchie standard for the C language [1]. It does not support port access functions. The AFSERVER program, though, does support port access.

The documentation for Parallel C, although reasonable when dealing with the implementation of the language, only briefly touches on the usage of AFSERVER: its principal aim is to allow the user to load and run the compiled C program. 3L do, however, supply information on request which describes how the undocumented Parallel C functions `_put_int()` and `_get_int()` can be used to construct port access functions [3].

Using this technique, the transputer board could read or write to the DSP's memory through AFSERVER. However, communicating with the DSP in such a manner requires 16-bit port accesses, and AFSERVER only supports byte access to the host's I/O ports. Hence two calls to AFSERVER must be made for each word read from the host.

Each block of data that must be read from the DSP per frame is 26 words long: one flag-word, a 16-bit word representing the short-time energy of the sampled frame,

and 8 coefficients each comprising 3 words.

A short benchmark program (`TIME_IO.C`) was written which timed 1000 such data transfers. The result was rather disappointing: the 1000 transfers took 54 seconds. Using such a transfer channel it would take 54 ms to transfer the information for one frame to the transputer, over five times that of the frame duration!

The main loss of time in this communications method can be attributed to the following factors:

- the access mechanism (byte or word) to the ports supported by `AFSERVER`;
- the extensive request protocol between `AFSERVER` and the transputer for each `AFSERVER` command.

For each request to read an I/O port the transputer must first send the request number, 38 for a port read [2], followed by the address of the port. The transputer must then read the value of the port followed by a status word which is sent by `AFSERVER`. Some time must also be lost by `AFSERVER` decoding the request number, and deciding that a port access was requested. This communications method is highly inefficient and is therefore not suitable for this application.

One possible way to reduce this inefficiency would be to have the PC read the ports into a block of memory, and then invoke a block transfer of the data to the transputer.

8.3.2 Stay-Resident Program to Facilitate Transactions

Parallel C offers routines to read blocks of memory from the host PC. Also supported is the ability for the transputer program to generate DOS interrupts via `AFSERVER`. One method of transputer-to-DSP communications utilising these facilities would have the transputer application call an interrupt service routine (ISR) which would read the relevant data from the DSP into a block of host memory. The transputer would then invoke a block read of the host's memory.

The ISR must be resident in memory when `AFSERVER` is running. One method of achieving this is to make the ISR part of a program (the parent program) which calls

AFSERVER as a child-process. Turbo Pascal has some useful procedures for creating interrupt-handlers and executing child-processes. Before executing AFSERVER, the parent-program must first set the vector of an unused DOS interrupt to point to the procedure that will act as the interrupt handler.

A Pascal program (PARENT.PAS) was written incorporating an interrupt handler that reads a block of data from the DSP into a block of 26 words of PC memory (see Section 8.3.1). This program invokes AFSERVER as a child-process and instructs it to load and invoke the transputer application CHILD.B4 (the compiled version of CHILD.C).

The transputer application invokes the PC interrupt-handler using the C function `int86()`. When the interrupt has been serviced, it reads the corresponding 26 word block of host memory. The program pair, PARENT and CHILD, time this process over 10^4 repetitions. The result at first glance was reasonably encouraging: 10^4 repetitions took 69 seconds. Therefore, one block of data could be read from the DSP in 6.9 ms.

On reflection, though, this is still far from ideal. For the duration of the data transfer, the transputer is solely tied with the communications, i.e. 69% of the processing time available on the transputer if real-time operation is to be achieved. With the inclusion of the analysis procedures timed in Section 8.2, the cumulative processor requirement would be 10.1 ms. This does not include the conversion of the DSP real-number format (3 words per number) into IEEE standard real format.

Although this time may well be acceptable (introducing only a delay of 1% of the utterance length) it is disappointing that the majority of processor time is required by the communications method. Transferring only 26 bytes from the DSP takes 6.9 ms when it only takes 3.1 ms to filter the LP coefficients, factorise the resultant LP polynomial, and track the pole-pairs. Obviously communicating through AFSERVER introduces major inefficiencies into the transfer method.

If real-time operation is to be achieved, allowing enough spare processing time for the future addition of other analysis procedures (such as perceptual mapping of the rs-plane) then it is necessary to develop a communications method that operates independently of AFSERVER.

8.3.3 Low-Level PC-to-T800 Board Communications

The transfer of information between the PC and the transputer application using the high-level procedures offered by `AFSERVER` is very inefficient due to its extensive communications protocol. To remove this inefficiency it is necessary to implement data transfers using low-level procedures. Such a communications method would involve terminating `AFSERVER` once the transputer application has been loaded, and communicating via the transputer/PC interface directly.

The interface comprises a C012 link adapter which is mapped into the PC's I/O space. The communications protocol with the link adapter is described in *The Transputer Databook* [5], and the register addresses in the I/O map are described in the *TMB08 Installation and User Guide* [4].

The Parallel C routines `_put_int()` and `_get_int()`, which are not documented in the Parallel C manual but covered in a 3L technical note [2], can be used to send 4-byte integers to `AFSERVER` using its *data protocol* [3].

The most straightforward data transfer system to implement would allow the transputer application to use these functions to send and receive data. A Module, `T800COMM.ASM`, containing two 8086 assembly language routines were written which read and write the data across the link adapter. Each call to these routines can transfer one data unit of size 32-bits (Parallel C types *int*, *short*, *long*, *float*, and *pointer*).

The IEEE single precision representation of real numbers is a 32-bit format. To decrease the communications time, the PC could read the three 16-bit words which represent a floating-point number on the DSP and convert them to the IEEE single format prior to transferring it to the transputer. Therefore, only one PC–transputer transfer would be required for each LP coefficient. In addition to the 8 LP coefficients, a flag must be transferred for each frame. This flag is used to tell the transputer when to stop expecting more data. Each frame therefore requires 9 transfers.

To estimate the data transfer speed for this method, a pair of short test programs were written, one in Pascal for the PC (`TIME_LA.PAS`) and a corresponding program in C for the transputer (`READ_LA.C`). These programs time 10^4 repetitions of sending nine 32-bit values across the link. The result was 10^4 repetitions took 9.44 seconds;

therefore, the data for one frame could be transferred in just 0.94 ms.

This communications method is therefore acceptable provided the PC could read a frame of data from the DSP and convert the DSP format floating-point numbers into single-precision format in under 9 ms. To verify this, a further Pascal test program was written (`TIMECONS.PAS`) which timed 10^4 repetitions of data input for one frame from the DSP, and 10^4 repetitions of the construction of 8 single-precision numbers from the DSP format.

The result was that a block of data from the DSP could be read in 0.51 ms and converted in 2.52 ms. That gives a total PC processor time of just over 3 ms required to read and convert the LP data for each frame: this process does not affect the transputer in any way. This communications method, therefore, is greatly superior to the two methods mentioned previously.

8.4 Inter-Process Dependencies

With the DSP module and the transputer module communication via the PC using the method described in Section 8.3.3, there are four processes that run concurrently, namely:

1. sampling the speech waveform on the DSP;
2. generating the LP coefficients on the DSP;
3. reading the LP coefficients from the DSP, converting their format, and sending the results to the transputer;
4. smoothing the LP coefficients, determining and tracking the pole-pairs on the transputer.

These processes constitute a pipeline system, with the output of each process passed on to the next until the pipe-tail (the transputer) is reached.

The first process in the pipeline is the sampling of the speech waveform. This process, implemented on the DSP, fills up a frame-buffer of 100 samples. Once the buffer is full, the sampling process resets its pointer to the beginning of the buffer.

Since the sampling process must take samples at regular intervals it cannot afford to be dependent on any other process: once it has received instructions to start sampling, it will sample once every 0.1 ms regardless of whether or not the other processes are ready for the information.

The second process in the pipeline is the LP analysis module implemented on the DSP. This waits until the sampling has been completed before copying the full buffer into a working buffer. 8th-order LP analysis is then applied to this frame of speech data. This analysis must be completed before the next frame of data has been captured i.e., within a 10 ms interval. The LP analysis, therefore, is dependent on the sampling process.

The LP analysis might also be dependent on the succeeding processes. If these processes have not completed analysing the results from the previous frame when the LP analysis is ready to pass on new data, then it would have suspend to operation until the data could be transferred. This is not a problem, provided it can be guaranteed that the remaining processes could complete their analysis within the allotted 10 ms time-frame. However, the transputer employs an iterative root-finding procedure, the duration of which cannot be accurately predicted. Thus, root-finding may take longer than the allotted 10 ms. If this were to happen it would miss the completion of the LP analysis for a particular frame.

To overcome this problem, the DSP-to-PC communications channel is buffered. This buffer, henceforth referred to as the LPC buffer, resides in the DSP's memory and has the capacity to hold a validation flag, an energy measure, and the eight LP coefficients for 250 analysis frames i.e. 2.5 seconds of speech.

The third process is implemented on the PC. Its task is to monitor the LPC buffer until a new set of data has been placed in it by the LP analysis module. It then reads the energy measure and LP data into the PC's memory. If the energy measure is the absolute maximum of `0x7FFF`, then it is reasonable to assume that the DSP's accumulator has saturated during its production. It is also reasonable to assume that the accumulator will saturate during the production of the covariance parameters and that the resultant LP coefficients are inaccurate. If this is the case then the attenuation of the input signal is increased by 2 dB and the data from the

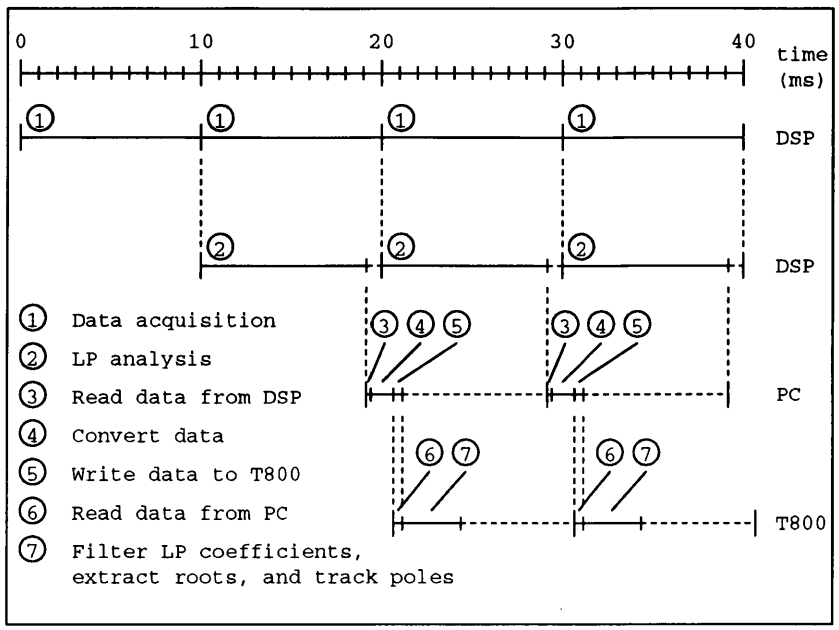

Figure 26: Process timings and inter-process dependencies represented on a Gantt-Chart.

frame is discarded.

If the energy measure is acceptable, then the LP coefficients that have been read in the DSP's 3-word format are converted to IEEE single-precision format and transferred to the transputer. This monitor/transfer process is therefore dependent on both the LP analysis and the process implemented on the transputer.

The transputer process is the pipe-tail. Its function is three-fold. Firstly it applies a smoothing function to the LP coefficients. It then determines the quadratic factors of the LP polynomial, and subsequently tracks the resultant pole-pairs. This is dependent on the monitor/transfer process on the PC.

Figure 26 shows the relative times of the jobs done by the different processors in the form of a *Gantt-Chart*. The dotted horizontal lines indicate processor idle time.

Data-acquisition will always take 10 ms for each frame. However, due to the time-slicing approach, it does not take up 10 ms of TMS32020 time: it actually takes up about 0.5 ms total processor time.

The LP analysis takes approximately 9.2 ms. Again, this time does not reflect the amount of processor time used: it takes about 8.7 ms of processor time.

Reading the coefficients from the DSP board, the first task of the PC, takes about 0.51 ms. The conversion of the LP coefficients from DSP format to IEEE single-precision format takes approximately 2.52 ms. The final task of the PC is to transfer the coefficients to the transputer, and takes 0.94 ms. The total time required by the PC is therefore 3.97 ms.

The first task for the transputer is to read the LP coefficients from the PC. This is exactly synchronised with the PC and therefore takes 0.94 ms. Its next task is to filter the LP coefficients, taking approximately 0.63 ms. Its penultimate task is to factorise the LP polynomial. As this is an iterative process, its timing cannot be accurately predicted, but is typically in the order of 1.6 ms. The final task of the transputer, tracking the pole-pairs, takes 0.91 ms. The typical time that the transputer is occupied is therefore in the order of 4 ms.

8.5 Possible User Feedback Modes

The simulation of the Tutor catered for one possible feedback mode: the extraction of the vowel portion from the utterance could only be done once the utterance had been completed. Hence the feedback concerning the quality of the vowel must be deferred until the end of the utterance giving rise to the *deferred feedback* mode.

However, the system is capable of analysing the speech waveform in real-time. Hence it should be possible to offer *immediate feedback* to the patient provided the host PC has enough processing power to govern the DSP-to-T800 communications *and* display results continually.

For both modes, the LP model for the speech waveform is determined by the DSP software. In deferred feedback mode, the transputer must smooth the LP coefficients, obtain the poles of the system, track the pole movements, identify the steady-state regions of the utterance, generate a template which represents the steady-state region, and finally match all target templates to the utterance template.

In immediate feedback mode, the transputer must smooth the LP coefficients and, when necessary, obtain the poles of the system and match the results against the target templates. The transputer must pass back these results at regular intervals

throughout the utterance production.

The speech is analysed at a rate of 100 frames per second. For animation purposes, it is only necessary to display results at a rate of approximately 20 times per second. To build some flexibility into the system, the choice of when feedback is required during the utterance is determined by the PC software. The PC program passes the LP coefficients of each frame to the transputer together with a flag indicating whether or not feedback is required. For this application, a flag with every 5th frame is set to indicate to the transputer that feedback is required.

8.6 Conclusions

A PC-hosted transputer package has been developed specifically for integrating with the LP analysis software described in Chapter 7. This package will read the LP coefficients produced by LP analysis software, apply a smoothing function to these coefficients, and determine and track the pole-pairs of the resultant LP system function.

Under normal circumstances, i.e. when analysing steady-state or slowly-varying voiced speech, the transputer application will operate in real-time. (In fact under these circumstances the transputer is idle for about 60% of the time.)

The next chapter describes how this package, combined with the LP analysis routines, has been integrated with a user interface to produce a speech therapy aid for vowel production.

8.7 References

[1] B.W. Kernighan and D.M. Ritchie (1978) *The C Programming Language*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ.

[2] 3L Ltd (1988) "Accessing host i/o ports using parallel c v2.0", Technical Report 2, 3L Ltd, Peel House, Ladywell, Livingston EH54 6AG.

[3] 3L Ltd (1988) "File service protocol definition", Technical report, 3L Ltd, Peel House, Ladywell, Livingston EH54 6AG.

[4] Transtech Devices Ltd, *TMB08 Installation and User Guide*, Transtech Devices Ltd, Unit 17, Wye Industrial Estate, London Road, High Wycombe, Bucks. HP11 1LH.

[5] INMOS Limited (1989) *The Transputer Databook*, INMOS Limited, 1000 Aztec West, Almondsbury, Bristol BS12 4SQ, UK.

Chapter 9

The Speech Therapy Tutor

Overview

This chapter describes the operation of the Tutor from a user's point of view. The visual aspect of the feedback modes is discussed in detail, together with the user interface to a simple database which is used to store template patterns. Also described is a pilot trial the purpose of which was to verify the operation of the Tutor.

9.1 Introduction

The previous chapter mentioned how two methods of feedback to the patient could be made available, namely *deferred feedback* and *immediate feedback*. The deferred feedback mode returns the result of the analysis at the end of the utterance, whereas the immediate feedback mode displays results as the utterance is being produced. This chapter describes how these modes were realised in the Tutor and how they were incorporated into a menu-driven environment.

As the patient uses the Tutor, the vowel portion of his or her (monosyllabic) utterance is matched against each member of a set of prototypes which have been chosen by the therapist. Ideally, these prototypes would be obtained by the therapist and patient working with the Tutor in learn mode, the therapist indicating to the computer when a good example has been spoken. Alternatively the prototype can be obtained from a tape recording of the therapist and patient working together and the identified satisfactory utterance used as the model.

The set of prototypes is stored as a database on the computer's hard disk. The composition of the database is flexible in that it could contain an example of one vowel for many speakers, or examples of several vowels by one or more speakers. The prototypes need not be examples of good quality utterances. The database could comprise examples of varying degrees of quality: for example if a database contained templates of good, medium, and bad utterances then the Tutor could indicate the target template to which the current template matched, and hence the quality of the utterance.

9.2 User Feedback Modes

In its present form, the system contains two modes of operation, namely the *deferred feedback* and the *immediate feedback* modes [1]. The former mode is used when the therapist requires that the vowel appear in a CV or CVC context. In this case, the vowel portion must first be located and so feedback is deferred until the whole utterance has been produced (with the template matching suppressed, this mode is

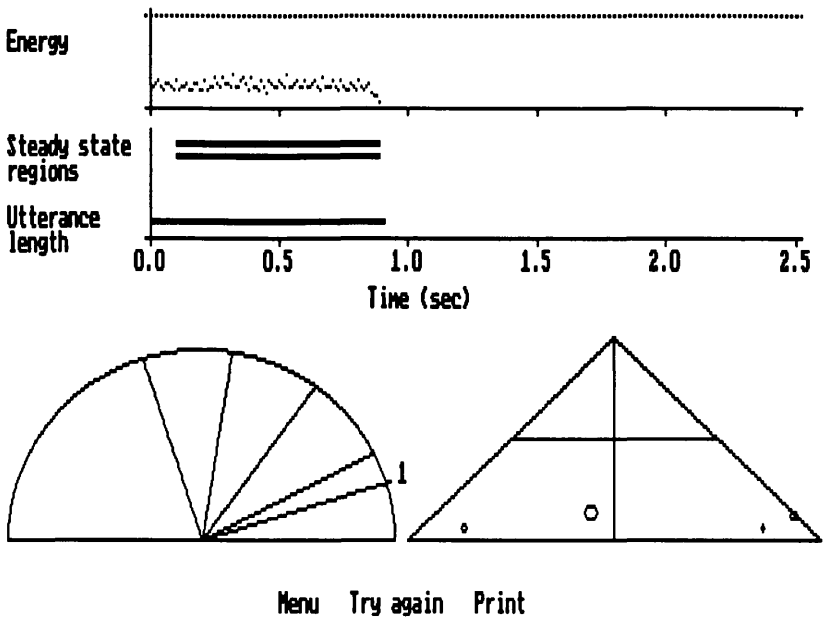

Figure 27: Typical screen display from the *deferred feedback* mode.

also used when generating prototypes).

Figure 27 shows a typical screen display after the analysis of an utterance using this mode. This display comprises five sections. At the top is a graph against time of the energy of the sound. Below this is a graph against time which shows the length of any steady-state regions of the utterance, and the length of the utterance itself. Underneath this graph are a 'dial' type display and a triangular display. The dial indicates the closeness of the utterance to a prototype (see Section 4.7). The triangular display shows the position of the pole-pairs on the rs-plane. The remaining section is the horizontal list of options at the bottom of the screen. As can be seen, the therapist can choose to exit this mode (return to the *Menu*), to continue in this mode and have the patient *Try again*, or to get a hard-copy *Print* of the current screen display. This list is only displayed at the end of the analysis of an utterance.

Prior to analysing the utterance, the Tutor will display the 4 plots devoid of any information and will bleep once to indicate that it is waiting for input. As the patient produces the utterance, the energy plot is updated. This gives the patient an indication of the length of his or her utterance so far. When the patient stops vocalising, the remaining information is displayed with no noticeable delay.

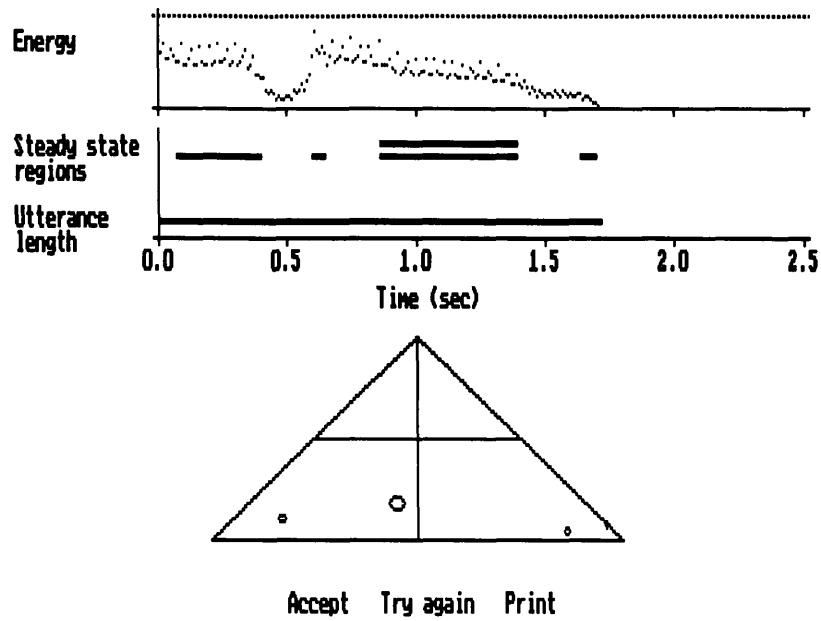

Figure 28: Typical screen display from the prototype generation mode.

The lower trace against time shows how steady the produced sound was. Ideally for a monosyllabic sound, with the vowel extended, this should be a continuous line which is slightly shorter than the length of the utterance. If the sound was not particularly steady then this line will be fragmented (or not present if the sound was transient throughout). The longest steady-state region is emphasised by being displayed as a double bar: this is the region that is taken for template matching. An example of this can be seen in Figure 28. This is actually a screen shot from a session in which prototype templates were generated. Notice that the screen is similar to that for the deferred feedback mode but has no dial showing a template match.

The dial display gives an indication of how close the utterance was to the best-matching prototype. The pointer moves clockwise from the left. The left-most position indicates a very poor match, and the right-most position indicates a perfect match.

The other mode, immediate feedback, displays results in real time as the speech sound is being produced. Using this mode, patients can experiment with moving their tongue, jaw, and lips, while watching the display to see if their production is close to the target sound or not. This allows them to learn the correct vocal-tract shape for

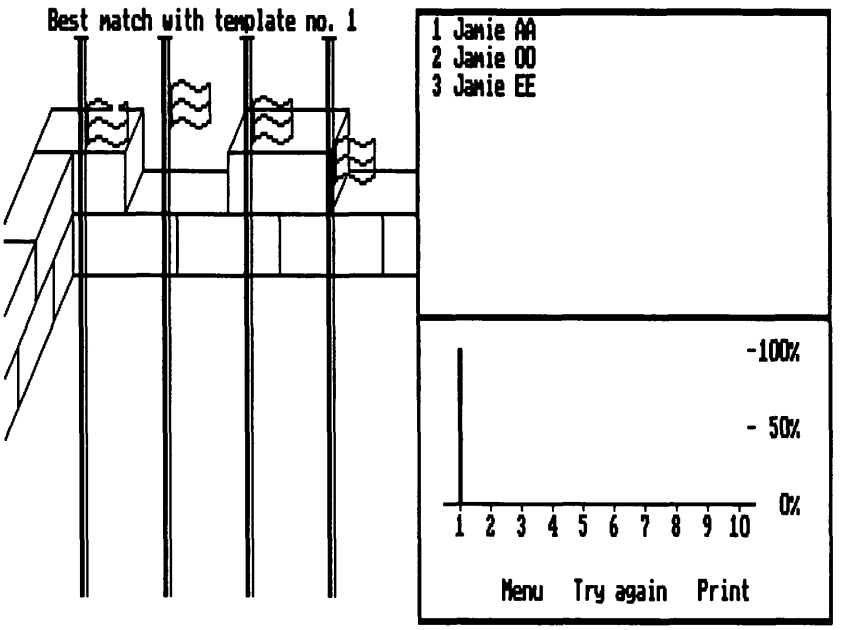

Figure 29: Typical screen display from the *immediate feedback* mode.

a particular sound experientially.

This mode has another possible use. As stated above, the deferred feedback mode is especially suited to vowels in a CV or CVC context provided the vowel sound is extended. However, there may well be patients who are incapable of extending such sounds. Using the immediate feedback mode the patient could practise making prolonged isolated vowels sounds and see for themselves when the sound is not steady.

Figure 29 shows a typical screen shot for the immediate feedback mode. Again there are five regions of displayed information. The largest region is that containing four flags on flag-poles which are situated on top of a castle tower. Each flag position gives an indication of the distance between a pole-pair on the reference template to the optimally matched pole-pair of the current 10 ms frame (see Chapter 3). Above these flags is a statement of which prototype template the sustained utterance matched closest to in any one analysis frame. The box in the top right of the screen gives a list of the name or identification of each entry in the database (in this case there are three entries). The box in the lower right of the screen displays a histogram of the performance throughout the utterance i.e. for each prototype a bar shows the number of frames which matched closest to it as a percentage of the total number of frames

of speech data analysed. This box also contains a list of options that the therapist has, which is identical to the list for the deferred feedback mode.

Prior to analysing the utterance, the screen is drawn as shown but with the flags at the bottom of each pole, no statement of best match, and no bars on the histogram. The Tutor then bleeps once to indicate that it is waiting for input.

As the sound is produced, it is analysed and matched against *all* templates in the current database and the flags indicate the difference between the utterance and the closest prototype template. As the utterance is being concluded, control over the articulators is relaxed, and consequently the flags will drop. If the final result is left displayed it is probable that this is not representative of the patient's overall performance. Therefore the best (highest) position of the flags is recorded and displayed at the end, giving a better indication of the patient's performance.

Once the flags have been repositioned, the statement (positioned above the flags) is made as to which prototype matched best and the histogram is drawn.

9.3 User Interface

The functions of the Tutor are all accessible via the main menu. This comprises a list of seven options. As well as an option to quit the Tutor, there are the two feedback modes described in the previous section, the prototype template generation option, *Add entry*, and three database management functions. The first database function, *Select database*, allows the user to change the database file. The second database option, *View names of entries*, is used to see the contents of the database at a glance. The final database option, *Edit entries*, permits the non-technical information for a database entry to be modified.

The desired action is chosen by one of two methods. Firstly, a cursor, which highlights one option in the list, can be moved up and down the list by using the corresponding arrow keys on the keypad. Once the desired action is highlighted, the <ENTER> key is pressed, and the action is taken. This approach is appealing for novice users [2]. An alternative menu selection mechanism is also implemented. Since each option has a unique starting letter, the cursor can be moved directly to an option

by pressing the corresponding letter on the keyboard.

The choice of current database from those available on disk is made in a similar manner to that of the first method described above. A list of all databases is shown on the screen and a cursor is moved until the desired database is highlighted. The cursor is moved horizontally by the left and right arrow keys and moved vertically using the up and down arrow keys. Once the chosen database is highlighted, it is loaded by pressing the <ENTER> key. Using this method of selection, the risk of the user typing the name of a non-existent database is removed. The user need only type the name of a database when he or she is creating a new one.

An on-line help facility has been incorporated for the main menu. Help on any of the menu options can be displayed by moving the cursor to the desired option and then pressing any of the function keys. The on-line help facility is also available when entering or editing a database entry. Each entry in the database comprises several fields of data, one for the patient's name, one for his or her age, etc. When an entry is displayed on the screen for editing each field is contained within a boxed text window. The text cursor is situated within the currently-selected text window. The cursor can be moved from one field to another as well as moved within the current window. There is a different help screen for each window and it is called into view by pressing any one of the function keys. Pressing any key when the help screen is displayed will return the user to the point that he or she was at, prior to calling up the help screen.

Many more features which the therapists might find useful could be built into the Tutor. These additional features include the ability to log the patient's performance throughout a therapy session, and to give the therapists the option of making a hard-copy of the results, both for themselves and for the patients: the therapists indicated that a hard-copy given to the patients can boost morale (provided the results are good).

These features would be reasonably simple to include in the Tutor. However, it is essential to let the therapists get a feel for what the Tutor can and can not do in its current state, prior to discussing with them what the exact form of these extras should be.

Database Reference	Vowel Identity
ADULT_V1	/a/
ADULT_V2	/o/
ADULT_V3	/i:/

Table 25: Vowel identity of each of the template databases.

For this purpose a preliminary operating manual has been written which has been given to the senior speech therapists together with the Tutor (the manual is included as Appendix D). It describes the operation and structure of the Tutor, as far as possible, in non-technical terms.

9.4 Pilot Verification Trial

A short pilot trial was devised to verify the overall operation of the Tutor and the suitability of the immediate-mode display as a feedback mechanism. The author created three databases each containing two instances of one extended vowel sound. These databases were referred to as ADULT_V1, ADULT_V2, and ADULT_V3 so as not to convey the phonetic identity of the vowel which each represented. The actual vowel, for which each database was a template, is shown in Table 25.

A number of speakers were given the task of identifying the vowel represented by the database. This was achieved by setting the Tutor in immediate feedback mode with one of the un-identified templates. The subject was allowed to experiment freely with as many different vowel sounds as he desired. Using the flags to obtain feedback he was told to decide, in his own time, which vowel he considered he was being trained to produce. His comments and oppinions were recorded by a supervisor.

Six cases are now reported. In the first two the procedure was supervised by the author (who was aware of the database content). In the remaining cases the supervision was by a person who was totally unaware of the vowel being trained.

The first subject, S1, was very co-operative. The first template that was loaded was ADULT_V2. After produced the sounds /a/, /o/, /u/, /e/, and /i:/ S1 decided

that /o/ was the best candidate. The second template that was loaded was `ADULT_V1`. S1 identified /a/ as the best candidate although to begin with /a/ did not show a particularly good match, he could try variations of /a/ and get the flags to move upwards. He could not move the flags upwards with other variations of the other vowels (one or more of the flags persistently stayed at the bottom of the display). The final template, `ADULT_V3`, was loaded. After trying several vowel sounds, S1 had no doubt that /i:/ was definitely the identity of the template.

The second subject, S2, was initially willing to take part in the trial, but soon became reluctant when he was positioned in front of the Tutor. Database `ADULT_V1` was loaded first. S2 sequenced through the vowels /e/, /u/, /o/, /i:/, and /a/. None of the sounds raised all of the flags close to the top, and hence S2 was reluctant to make a decision as to which vowel the template best represented. When it was explained that it was not the absolute positions of the flags that was important in this trial, but was a comparison of the results of each vowel to determine which vowel was *best* represented by the template, S2 decided that /a/ was the best candidate. The second database to be loaded was `ADULT_V2`. In this case it was difficult to choose between the results of vowels /i:/ and /o/: both had 2 flags at mid-level with the other two at the lowest position, and the subject did not make a final decision. The third database to be loaded was `ADULT_V3`. For all of the vowels tried, all had at least one flag at the lowest position. It was decided that /e/ was probably the best candidate, however this decision may have been influenced by the supervisor who knew the identity prior to running the trial. S2 tried database `ADULT_V2` again and came to the conclusion that /o/ could be favoured as best candidate. Again influence from the supervisor may have been present.

It was possible for the supervisor to influence the decision of the subject, either wittingly or not, if he knew the identity of the vowels that the databases represented. Therefore, it was decided that a new supervisor would be elected who did not know this information but understood the operation of the Tutor.

Four more subjects, S3 to S6, were found to participate in the trial with this new supervisor. Table 26 shows the results of the whole trial. For each database there are two columns. The leftmost is the vowel that was ultimately chosen as the

Subject	Database Reference					
	ADULT_V1		ADULT_V2		ADULT_V3	
	1st	2nd	1st	2nd	1st	2nd
S1	/a/	–	/o/	–	/i:/	–
S2	/a/	–	/o/	i:	/i:/	–
S3	/a/	/o/	/o/	/a/	/i:/	–
S4	/0/	/a/	/o/	/0/	/i:/	–
S5	/a/	/o/	/o/	/a/	/i:/	–
S6	/a/	/0/	/o/	–	/i:/	/u/

Table 26: Results of Pilot Trial.

most probable identity of the database. In some cases, the subject narrowed the possibilities down to two vowels before finally deciding on the best candidate and the vowel rejected is shown in the other column. Note that due to a lack of fonts for the document preparation system, the author has adopted the symbol /0/ to represent the first vowel in "bottle".

It can be seen that in all cases except database ADULT_V1 for subject S4, the subject was successful in determining the identity of the vowel which the database represented. It can be concluded, therefore, that the Tutor is able to discriminate vowel sounds and that the immediate-mode is a suitable feedback mechanism.

9.5 Conclusions

A working prototype of the Tutor for vowel production has been produced. It analyses the speech waveform in real-time and offers two modes of feedback to the user. One mode displays information as the utterance is being produced and the other allows the extended vowel to be in a consonant-vowel or consonant-vowel-consonant context and defers feedback until the end of the utterance.

For both modes, the utterance is matched against all templates within a database of prototypes, the structure of which is flexible. The modes are available from a main menu together with some database management functions.

A short pilot study was carried out which verified the operation of the Tutor and the suitability of the immediate-mode as a feedback mechanism.

9.6 References

[1] J.M Turnbull, A.T. Sapeluk and R.I. Damper (1990) "A new computer-based speech therapy tutor offering immediate and deferred visual feedback", *Proceedings of the Institute of Acoustics, 1990 Autumn Conference, Speech and Hearing*, **12**, 63–70.

[2] B. Shneiderman (1987) *Designing the User Interface*, Addison-Wesley, Reading, MA, 117–120.

Chapter 10

Results of Field Testing

Overview

This chapter details the evolution of the Tutor as a direct result of suggestions made by the therapists. These suggestions affect both the cosmetic appearance of the Tutor and its functionality.

This chapter also highlights the views of the therapists regarding the usefulness of the Tutor as a speech therapy tool. These views stem primarily from a short pilot study with a post-pharyngoplasty cleft palate patient.

10.1 Introduction

To enable field testing of the Tutor in clinical applications, two prototypes were constructed. One was incorporated into a portable PC, and the other utilised a desk-top PC. The portable Tutor was given to therapists for their use and the desk-top version was used to develop further or modify the Tutor's software.

The first task assigned to the therapists was to experiment with the Tutor by themselves, without patient contact. This accomplished three objectives. First, the therapists would test the Tutor's operation, reporting any *bugs*. Second, the therapists would learn how to use the Tutor. Third, the therapists could suggest alterations to the Tutor if it was felt that it was lacking in some aspect.

Meetings were arranged at intervals of approximately two weeks, during which the therapists could discuss the operation of the Tutor and highlight any pros and cons with regard to the functionality or user interface.

Once the therapists were satisfied with the Tutor's operation and desired alterations had been implemented, the Tutor was subjected to a short pilot study. The remainder of this chapter details what alterations were suggested by the therapists and the results of this study.

10.2 Alterations Suggested by Therapists

One of the first suggestions made by the therapists was the inclusion of an on-line help facility. This was implemented in two stages. First, it was requested that on-line help should be included in the database entry screen as a matter of priority: this is where the manual (see Appendix D) was consulted the most. Second, on-line help was incorporated into the main menu, so that a summary of each function could be viewed by the novice user. The therapists also pointed out that the only way of telling which database had been loaded was to add an entry to it; the top left of the information entry screen displays this. The Tutor was thus modified so that the current database is displayed at the top left of the main menu screen.

An early modification was to the option selection for the three modes *Add entry*,

Deferred feedback, and *Immediate feedback*. A poor choice of command words generated some confusion over what option to use. All three modes had the options *Quit* and *Run* (this made the programming easier: one procedure was used to cover all three cases). *Quit* was used to exit that mode to get to the main menu for the two feedback modes or to get to the data entry screen in the *Add entry* mode. *Run* was used to re-run that particular function. This is fine for the deferred and immediate feedback modes. The problem arises in the add entry mode: intuitively, quit is a negative action so should not be used to *accept* the template. For all three modes, *Run* is replaced by *Try again*. For the deferred and immediate options *Quit* is replaced by *Menu*. For the add entry mode *Quit* is replaced by *Accept*.

It was also suggested that hard-copy printouts should be made available for the above modes. A *Print* option was added to all three.

For the non-immediate feedback modes, the plots against time have changed markedly from those for the first version. Initially there was no energy plot and the steady-state region indicator was a very small plot in the top left of the screen (it was a left-over from a debugging session). It turned out that the therapists liked the plot and felt that it should be given more emphasis.

Some changes were also suggested for the immediate feedback mode. In the first version of the Tutor, the information displayed was the closeness of match using the four flags on the flag-poles. For cosmetic purposes, these poles were situated on the top of a castle rampart together with a cannon. The cannon, though, had no function other than it helped *set the scene*. It was the therapists' view that this non-functional object was a waste of space and that other, more useful, information could be displayed in its place. The names of the entries in the database and the histogram now take its place.

Another modification to the immediate feedback mode was suggested. Initially, when the utterance was finished, the flags showed the final template match. It was felt that it would be better if the best, instantaneous, match was displayed at the utterance's conclusion (see Section 9.2).

10.3 Results of a Short Pilot Study

The therapists involved in field testing the Tutor indicated that the client group, in Tayside's Region, requiring therapeutic vowel modification who would respond to a biofeedback method is very small. To date, it has only been possible to carry out one short pilot study with a client (see later on in this section). Speech therapists working with cleft palate, hearing impairment, dysphonia and dysarthria could all identify occasions when the Tutor would have application, but these were not represented in their ongoing caseload at the time of the pilot study.

The speech therapists also considered the specific use of the Tutor and its appeal to their clients. It was felt that the current format would be more suitable for adults, and that colourful and animated displays would have greater appeal to children.

Discussions with the therapists resulted in modifications and simplifications to the display as described in the last section. Although it was felt that the information presented was clearer as a result, it was pointed out that the displays needed to be almost self-explanatory: it must be remembered that many speech therapy clients have additional physical, visual, and/or intellectual impairments.

The therapists were impressed with the flexibility of database structure the Tutor offered, e.g. for a client's range of vowels; for any vowel with a range of speakers; for an individual production of a repeated vowel; and so on. Although the Tutor retains these prototypes, it was felt that auditory reproduction (by storing and replaying the actual digitised utterance used to create the prototype) would be required to allow full use of this facility: it is very easy to forget what the actual prototype sounds like.

In the pilot study, a post-pharyngoplasty[1] cleft palate patient, a sixteen year old girl, used the Tutor weekly for 5 weeks in October and December 1989. Each session was 30 minutes in duration and the Tutor was used for about 20 minutes each time. A database consisting of her vowels was created and used to encourage consistent good production, matched to a previously agreed prototype. The girl enjoyed using the Tutor and it helped motivate her and thus facilitated treatment. She was at a plateau of improvement and the Tutor was used by the therapist to convince the patient that

[1]Secondary cleft palate operation performed by plastic surgeons in order to change the shape of the soft palate.

she could produce and maintain consistent vowel sounds.

10.4 Conclusions

From the therapists interaction during the latter development stages of the Tutor and the short pilot study it has been seen that the Tutor has several positive features:

- detailed information is available on the display;
- it has a flexible database structure;
- clients enjoy using 'technology'.

The Tutor also has some negative sides that have been identified:

- it addresses the needs of a very small client group, at least local to Dundee;
- the display and graphics require further simplification and increased appeal, especially for children;
- there is a need for auditory feedback to remind users what sound they are aiming for.

Possible solutions to these problems are suggested in Chapter 11.

Chapter 11

Future Work and Conclusions

Overview

A computer-based speech therapy tool for aiding the speech therapist in the correction of vowel production difficulties has been proposed. This chapter highlights the *pros and cons* of the Tutor.

Also in this chapter is a section suggesting various avenues which could be explored to continue the research.

11.1 Future Work

There are various avenues which could be explored to enhance aspects of the Tutor, and to further the research into tracking the pole-pairs on the rs-plane. These include the following:

- a clinical trial of the Tutor with a long development cycle;
- possible methods for making the Tutor more appealing to both the therapist and the patient;
- extension of the facilities offered by the tutor;
- reduction of the complexity and cost of the implementation by developing a single plug-in card for the PC;
- a study of the equivalence of the pole-pair tracking technique to conventional formant tracking techniques;
- research into methods for extending the tracking technique to cope with dynamic sounds such as short vowels and semivowels.

11.1.1 Clinical Trials of the Tutor

The speech therapy tutor was developed with help and advice from Dundee Speech Therapy Service, Tayside Health Board, with the majority of their input during the design of the Tutor's user interface. It was possible for a short pilot study of the Tutor to be carried out in a clinical environment. However, it was only possible for this study to be carried out with a *single* patient. To make an accurate assessment of the Tutor's capabilities and suitability in a speech therapy environment it will be necessary to carry out a clinical trial with a long development cycle.

11.1.2 Improving the User Interface

To make the Tutor more appealing to both the patient and the therapist, the visual displays must be re-done with hi-resolution colour graphics. It was probably a mistake

to use the LCD display in the first place. However, there had to be an early decision made between a portable monochrome-LCD machine and a bulky colour machine.

The decision was made to take the portable option for security reasons but in hindsight it is the author's view that this was wrong. A colour machine can be securely fitted onto a trolley and hence it could be wheeled into a locking cupboard at the end of the day. Alternatively, the price of portable machines which use colour-VGA resolution liquid crystal displays will inevitably drop in the future.

More imagination and research needs to be put into the visual feedback displays. One possible method of increasing the appeal of the Tutor would be to offer a choice of display. For example, an alternative choice to the four moving flags in the immediate feedback mode could be four children holding balloons. The balloons would move up and down depending on the closeness of match. These alternative displays could be geared towards different age groups, and possibly different sexes.

11.1.3 Extended Functions for the Tutor

One problem of the Tutor which became apparent during the clinical trials is that it addresses the needs of a very small client group. To increase the justification for having the Tutor, it should offer more functions such as games or tools which incorporate voiced/unvoiced decisions or pitch (such as in VisiSpeech for the BBC Microcomputer and SpeechViewer for the IBM PS2). Another display could show frequency/amplitude comparisons (using FFTs) between the target and the current speech pattern (such as in SpeechViewer). It may also be beneficial to have a tono-topical frequency scale as opposed to the linear scale used in SpeechViewer (on which equal frequency intervals on the display are not perceived as being equal). A real-time display of the vocal tract should also be a possible extension.

With an increased number of possible displays, it would be sensible to incorporate the tools into a windowing environment such as Microsoft's Windows 3. Each tool would have an associated icon and be selected using a mouse and pointer. The therapist could save particular set-ups for a therapist/client combination which could be re-called at the start of each session. Programming for Windows has a reputation of being extremely difficult and expensive. Until recently, the only tools available

which used a common programming language was Microsoft's *Software Development Kit* (SDK) for windows. This package had the added disadvantage that it can only be used with Microsoft's own C compiler. An equivalent alternative has been launched by Borland. This package, *Borland C++*, has the advantage that (at the time of launch) it is approximately half the cost of Microsoft's combined SDK and compiler and has the added advantage that it is also a C++ compiler (which has the potential of easing the workload of the programmer once a comprehensive set of Windows object classes have been defined). Soon after the release of this product, Borland launched an object-oriented Pascal compiler which has precompiled *units* for programming Windows applications (it is also cheaper than the C++ package).

It may be worthwhile investigating whether there are non-speech-therapy applications for this product. For example, the Tutor should be applicable to foreign language training.

11.1.4 Other Speech Sounds

The pole-pair tracking technique that has been presented was shown to overcome problems associated with formant tracking i.e. the varying number of formants problem. To establish the rs-plane solution as an equivalent alternative, it should be compared to spectrograms of continuous speech (which is normally done to verify formant tracking techniques).

The Tutor at present is quite restricted in that it will only deal with extended vowels. In this case, the pole-pairs reach a steady-state position. The above study would determine if it is possible to track the pole-pair movements during short (dynamic) vowels in order to project the trajectories to estimate the steady-state positions that would have been reached had the vowel been extended.

It should also be possible to use the tracking technique to map the time evolution of the pole-pairs during production of the dynamic semivowels. These traces could be compared to templates using a dynamic programming algorithm.

11.1.5 Cost Reduction

Methods of reducing the cost of the system should also be addressed. It could be possible to replace the TMS32020 plug-in card with a card which utilises the SP1000 LP analysis chip from General Instruments. This chip produces the reflection coefficients of the acoustic-tube model and hence they would have to be converted to the LP coefficients (see [6]). The present system also uses a single transputer module (tram) on a mother-board which can hold 10 trams. This could be replaced with a single transputer card.

Another solution would be to use one of the more powerful digital signal processors that have recently been introduced (and hence were prohibitively expensive at the time of developing the Tutor). One such processor is AT & T's DSP32C. This is a floating-point digital signal processor which is capable of 25 MFLOPS (actually 12.5 million floating-point multiply/accumulate operations per second). It has an operating speed of 50 MHz giving the time of each machine state of 20 ns (if zero wait-state operation is achieved then one instruction cycle is 4 machine states). The DSP32C has a 32-bit architecture and the floating-point representation is equivalent to the IEEE single-precision format. The same number of bits are used for the mantissa and the exponent; however, the DSP32C uses a two's complement representation for the mantissa whereas the IEEE single precision format uses a positive mantissa with a sign-bit.

The covariance method of LP analysis was written in assembly language for this processor. Using an 8th-order model and a frame size of 100 samples the DSP32C can perform the LP analysis in under 200 μs. The actual time taken was 198.88 μs and was measured for both program and data in zero wait-state RAM using the software simulator which comes with AT&T's software support library.

This time could be reduced if two copies of the frame of data were to reside in separate banks of zero wait-state RAM: the DSP32C will insert one conflict wait state (20 ns) if two consecutive memory accesses are addressed to the same physical memory block (of which it has 4; 3 constitute internal memory and 1 constitutes all external memory).

The DSP32C (and its predecessor the DSP32) mnemonics and assembler have the

appearance and feel of a high-level language similar to C. This aside though, the DSP32C is quite a tricky processor to program: due to the pipeline architecture of the device, it is possible for the result of some calculations not to be available until 3 or 4 instructions later. This effect is called instruction latency (see [2]) .

For example, if an accumulator is used as an input to the multiplier of the data arithmetic unit, then it must be established 3 or more instructions previously. Consider the following code fragment:

```
a0 = *r1        /* assume that r1 points to a memory location  */
                /* which contains the number 2.0 i.e. a0 = 2.0 */
                /* assume that r2 points to a memory location  */
                /* which contains the number 4.0               */
a0 = a0 + *r2   /* a0 is now 6.0                               */
a1 = a0 * a0    /* a1 is now 4.0 not 36.0                      */
```

To get the desired result the code should have been as follows:

```
a0 = *r1        /* assume that r1 points to a memory location  */
                /* which contains the number 2.0 i.e. a0 = 2.0 */
                /* assume that r2 points to a memory location  */
                /* which contains the number 4.0               */
a0 = a0 + *r2   /* a0 is now 6.0                               */
nop
nop
a1 = a0 * a0    /* a1 is now 36.0                              */
```

The `nop` (no-operation) instructions are obviously a (sometimes necessary) waste of the processor's resources. In many cases these can be replaced with instructions that are pertinent to the following code (providing these interlaced instructions do not attempt to alter `a0` or `a1` in this case).

The final implementation was some 20% faster than the original direct translation of the TMS32020 code. This was achieved by noting that all divisions in the LP analysis procedure use a member of the vector `DVec` as a denominator. As this vector

is the same length as the model order, the actual number of divisions required in this case can only be at most 8 (as opposed to 36, see Section 7). The DSP32C floating-point unit can only perform addition, subtraction, and multiplication. It does not have a hardware implementation of floating-point division algorithm; hence, divisions have to be implemented in software (routines for which are available in [3]). For the DSP32C this saving in the number of divisions required is a very large saving indeed. Once all of the 8 reciprocals of `DVec` have been calculated, each division (of about 2 μs execution time) can be implemented as a multiplication (of 80 ns execution time). The saving on the TMS32020 would also be quite considerable. The 36 divisions (of execution time 18.4 μs each) can be replaced with 8 divisions and 36 multiplies (of execution time 9.2 μs). This would constitute a saving of $36 \times 18.4 - (8 \times 18.4 + 36 \times 9.2) = 184$ μs.

At the time of writing, the DSP32C processor could be purchased in bulk for around £70 each [4]. Design considerations for a plug-in card for PC compatibles are described in the Application Guide [5].

11.2 Conclusions

A computer-based speech therapy tutor for vowel production has been developed. It is based on a PC-AT compatible with added processing power in the form of TMS32020 and T800 transputer plug-in cards. The Tutor uses a novel method of pole-tracking to identify steady-state (prolonged) vowels within a CV or CVC context. It operates in real-time and offers deferred and immediate visual feedback.

The Tutor segments the speech waveform into 10 ms contiguous analysis frames. The coefficients of the LP model for speech production are estimated, using the covariance method, on the TMS32020 digital signal processor (see Chapter 7). The LP coefficients are filtered to smooth the frame-to-frame variations and ultimately produce smooth tracks (see Chapter 4). The quadratic factors of the LP polynomial are estimated using a modified version of Bairstow's method. These factors represent pole-pairs and can be shown as a single point on the rs-plane. For the 8th-order

model used, this gives four points on the rs-plane which are tracked in order to determine any steady-state regions in the utterance for the deferred feedback mode. For the immediate feedback mode the four points are mapped onto a more perceptually meaningful form of the rs-plane and matched against a set of reference templates (see Chapter 3).

On reflection, filtering the LP coefficients may not be the most desirable method of producing smoother tracks. Due to the highly complex non-linear relationships between the factors and the coefficients of the LP polynomial, direct processing of the coefficients cannot guarantee a stable filter as the end result (although there has been no evidence of such harmful effects during the development of the method). Filtering the coefficients was used as an alternative to using large frames which overlap to increase frame-to-frame correlation and hence produce smoother time variations. The problem with large frame sizes is the computation required to create the covariance parameters as the first step to obtaining the LP coefficients using the covariance method. However, as has been seen in Chapter 7, this process can be implemented on the TMS32020 using integer arithmetic and the `MAC` (multiply-accumulate) instruction. Since the remainder of the process is implemented using floating-point representation, the generation of the covariance parameters represents a small percentage of the overall computation, and hence an increase of the frame size is not as prohibitive as it might first appear.

One reason for not using the autocorrelation-method of obtaining the LP coefficients was the necessity for frame overlap due to the windowing of the frame of speech data. However, this might not have been a bad idea after all. Using the autocorrelation method might have been preferable due to the production of the reflection coefficients as a by-product of solving the set of linear-simultaneous equations recursively. These parameters could have been exploited to add a real-time area-graph display to the Tutor's facilities. Although this would not be original work it would have added to the functionality of the Tutor and may have resulted in more utilisation during the field trials (see Chapter 10).

Another alternative to filtering the LP coefficients is to filter the movements of the poles-pairs on the rs-plane. This could be done in conjunction with the increased

frame size and frame overlap. The increase in frame size and overlap would be used to ease the tracking process and once a pole-pair had been associated with a particular track a two-dimensional smoothing function would be applied. One possible candidate for this is the Kalman filter (see [1] for a good introduction to Kalman filters with relation to tracking).

A short pilot study of the Tutor has been carried out in a clinical application and the results were positive. Although there were some criticisms about the operation of the Tutor, the suggested alterations were mainly cosmetic. However, it was only possible for this study to be carried out with a *single* patient and to make an accurate assessment of the Tutor's capabilities and suitability in a speech therapy environment it will be necessary to carry out a clinical trial with a long development cycle.

With the incorporation of some other facilities (such as those described in the previous section) and with the visual-feedback displays based on a more appealing colour-graphics environment, the author is sure that the Tutor will be a useful speech-therapy tool for the correction of vowel disorders.

11.3 References

[1] A. L. C. Quigley (1972) "Tracking study: an introduction to the use of Kalman filters", *ASWE Technical Report TR-72-14.*

[2] AT& T (1990) *The WE DSP32C Digital Signal Processor Information Manual.*

[3] AT& T (1988) *The WE DSP32 and DSP32C Support Software Library User Manual.*

[4] Personal communication. Bytech Components Limited, 12a Cedarwood, Chineham Business Park, Crockford Lane, Basingstoke, Hampshire RG24 0WD.

[5] AT& T (1988) *The WE DSP32 DSP Application Guide*, 106–114.

[6] L.R. Rabiner and R.W. Schafer (1978) *Digital Processing of Speech Signals*, Section 8.8.6, 443–444, Prentice-Hall, Englewood Cliffs, NJ.

Appendix A

Complexity of the Key Procedures

A.1 Bairstow's Root-Finding Method

The main iterative loop of Bairstow's method is shown below, coded in Pascal. It has been split into four sections to ease the identification of the number of multiplicative operations (multiplications and divisions) required.

```
{Section 1}
QuotientCoeff[2] := PolyCoeff[2]+QuotientCoeff[1]*RFactor;
DerivCoeff[2] := QuotientCoeff[2]+DerivCoeff[1]*RFactor;

{Section 2}
FOR i := 3 TO PolyOrder+1 DO
  BEGIN
  QuotientCoeff[i] := PolyCoeff[i]+QuotientCoeff[i-1]*RFactor
                      +QuotientCoeff[i-2]*SFactor;
  DerivCoeff[i]    := QuotientCoeff[i]+DerivCoeff[i-1]*RFactor
                      +DerivCoeff[i-2]*SFactor;
  END;

{Section 3}
Determinant  := DerivCoeff[PolyOrder]*DerivCoeff[Polyorder-2]
                -DerivCoeff[PolyOrder-1]*DerivCoeff[Polyorder-1];
DeltaRFactor := (QuotientCoeff[PolyOrder]*DerivCoeff[PolyOrder-1]
                 -QuotientCoeff[PolyOrder+1]*DerivCoeff[PolyOrder-2])
                 /Determinant;
DeltaSFactor := (QuotientCoeff[PolyOrder+1]*DerivCoeff[PolyOrder-1]
                 -QuotientCoeff[PolyOrder]*DerivCoeff[PolyOrder])
                 /Determinant;

{Section 4}
RFactor := RFactor+DeltaRFactor;
SFactor := SFactor+DeltaSFactor;
```

The number of multiplications and divisions for each section are as follows, where n is the order of the polynomial for which a quadratic factor is being estimated.

Section	Number of multiplies and divisions
1	2
2	$4(n-1)$
3	8
4	0

Each iteration therefore requires $2+4(n-1)+8 = 4n+6$ multiplicative operations. It is necessary to make the assumption that the number of iterations required for each factor is constant, say I. This initially seems a rather rash assumption, but as can be seen in Table 4 on page 24 it is not too far out for the application for which the method was employed.

If the calculation of each quadratic factor requires I iterations, then to calculate all quadratic factors of a pth order polynomial would require the following number of multiplies, m, where $q = p/2$:

$$
\begin{aligned}
m &= \overbrace{I(4\times 4+6) + I(4\times 6+6) + \cdots + I(4\times p+6)}^{(p/2-1)\text{ items}} \\
&= 6I(p/2-1) + 4I(4+6+\cdots+p) \\
&= I(3p-6) + 4I(-2) + 4I(2+4+6+\cdots+p) \\
&= I(3p-14) + 8I(1+2+3+\cdots+q) \\
&= I(3p-14) + 8I(q^2+q)/2 \\
&= I(3p-14) + 4I(p^2/4+p/2) \\
&= I(3p-14) + I(p^2+2p) \\
&= I(p^2+5p-14)
\end{aligned}
$$

A.2 Cumulative Distance Measures

This section describes the number of arithmetic operations required in the calculation of the cumulative distance measure for the tracking process. The two metrics discussed are the Chebychev, or city block, and the Euclidean distance metrics.

When tracking the movements of q objects from one frame to another it is necessary to consider $q!$ permutations of possible correlations if the guaranteed optimal result is desired. For each possible permutation a *cumulative distance measure* must be calculated. The permutation which yields the smallest cumulative distance is chosen to be the optimum correlation. The cumulative distance is calculated as a sum of the individual movements of each object within a frame.

At first sight it might appear that this requires q individual distance calculations for each permutation, giving a total of $q \times q!$ distance calculations and $(q-1)q!$ additions to obtain the optimum result.

However, it can be seen that only q^2 individual distance need be calculated and placed in a look-up table: if q objects are considered then each object of the current frame is associated with q others from the previous, giving q^2 associations ($q!(q-1)$ additions are still required to calculate the cumulative distances).

For the Chebychev distance metric, the individual distance for the ith object, d_{ci}, is given by:

$$d_{ci} = |x_{2i} - x_{1i}| + |y_{2i} - y_{1i}|$$

This requires three additions for each association. The total number of additions required to complete a table of associations is therefore $3q^2$, and therefore the total computational overhead is $q!(q-1) + 3q^2$.

For the Euclidean distance metric, the individual distance for the ith object, d_{ei}, is given by:

$$d_{ei} = \sqrt{(x_{2i} - x_{1i})^2 + (y_{2i} - y_{1i})^2}$$

This requires three additions, two multiplies, and a square root for each association. Therefore the total computational overhead is $q!(q-1) + 3q^2$ additions, $2q^2$ multiplications, and q^2 square roots.

Appendix B

Computing the LP Covariance Matrix

$$\begin{array}{ccccccc}
\Phi_{1,0} & \Phi_{1,1} & & & & \\
\Phi_{2,0} & \Phi_{2,1} & \Phi_{2,2} & & & \\
\Phi_{3,0} & \Phi_{3,1} & \Phi_{3,2} & \Phi_{3,3} & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \\
\Phi_{8,0} & \Phi_{8,1} & \Phi_{8,2} & \Phi_{8,3} & \cdots & \Phi_{8,8}
\end{array}$$

Figure 30: Required covariance parameters for an 8th order LP model

The required covariance parameters for an 8th order LP model are shown in Figure 30. $\Phi_{i,j}$ is calculated using the modified autocorrelation function on a N-sample segment of speech data, s, such that

$$\Phi_{i,j} = \sum_{m=0}^{N-1} s(m-i)s(m-j) \tag{17}$$

If this equation was used to calculate all 44 elements, $44(N-1)$ additions and $44N$ multiplications would be required. For large N this is quite prohibitive. However, it can be shown that there is a relationship between diagonally adjacent elements. Manipulating Equation 17 gives

$$\begin{aligned}
\Phi_{i,j} &= \sum_{m=0}^{N-1} s(m-i)s(m-j) \\
&= \sum_{m=1}^{N-1} s(m-i)s(m-j) + s(-i)s(-j)
\end{aligned}$$

and also substituting $i-1$ and $j-1$ for i and j respectively gives

$$\begin{aligned}
\Phi_{i-1,j-1} &= \sum_{m=0}^{N-1} s(m-i+1)s(m-j+1) \\
&= \sum_{m=1}^{N} s(m-i)s(m-j) \\
&= \sum_{m=1}^{N-1} s(m-i)s(m-j) + s(N-i)s(N-j)
\end{aligned}$$

And hence the relationship between diagonally adjacent elements is found to be

$$\Phi_{i-1,j-1} = \Phi_{i,j} - s(-i)s(-j) + s(N-i)s(N-j) \tag{18}$$

Consider again Table 30. Equation 17 could be used to calculate the 9 elements $\Phi_{8,0}$ to $\Phi_{8,8}$. The next 8 elements $\Phi_{7,0}$ to $\Phi_{7,7}$ would be calculated from the first 9 using Equation 18. The next 7 would be calculated from these 8 and so on down to the last two elements $\Phi_{1,0}$ and $\Phi_{1,1}$.

The first 9 elements would require $9(N-1)$ additions and $9N$ multiplications and the remaining 35 elements would require a total of 70 multiplications and 35 additions and 35 subtractions.

Appendix C

Perceptually Warped rs-plane

An analytical expression for the conversion of frequency, f, (in Hz) into critical-band rate, x, (in Bark) is given by

$$X(f) = \frac{26.81 \times f}{1960 + f} - 0.53 \tag{19}$$

It is desired that the scaling on the $s = -1$ line on the rs-plane is proportional to this within the range $r = -2$ to $r = 2$. $X(0)$ maps to the point $[2, -1]$ and $X(F_s/2)$ maps to the point $[-2, -1]$, where F_s is the sampling frequency.

The argument, θ, of a vector drawn from the origin and a pole (which will be one of a complex-conjugate pair) on the z-plane relates to the real frequency of the corresponding pole in the following manner:

$$\theta = \frac{f}{F_s} \times 2\pi$$

Re-arranging this equation gives

$$f = \frac{\theta \times F_s}{2\pi} \tag{20}$$

To obtain the equivalent of Equation 19 which maps the pole frequency, θ, onto the Bark scale, Equation 20 is substituted into Equation 19 giving

$$\begin{aligned} X'(\theta) &= \frac{26.81 \times \frac{\theta \times F_s}{2\pi}}{1960 + \frac{\theta \times F_s}{2\pi}} - 0.53 \\ &= \frac{26.81 \times \theta}{1960 \times \frac{2\pi}{F_s} + \theta} - 0.53 \end{aligned} \tag{21}$$

The scaling function for the rs-plane dimension can be obtained from $X'(\theta)$ through a series of transformations and dilatations such that $X'(0)$ maps to $B(0) = 2$ and $X'(\pi)$ maps to $B(\pi) = -2$ (see Figure 31).

The scaling factor of the dilatation is the ratio of the two respective lengths, i.e. $-4/\Delta X'$ where $\delta X'$ is $X'(\pi) - X'(0)$. The offset of $X'(0)$ is removed prior to scaling and an offset of 2 is added subsequent to scaling.

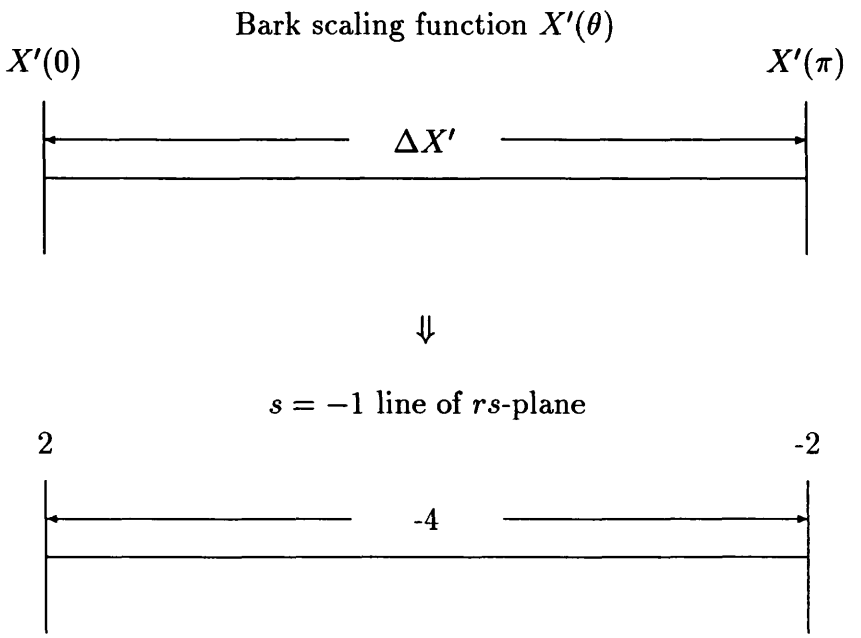

Figure 31: Mapping of Bark scaling function $X'(\theta)$ onto the $s = -1$ line of the rs-plane

The resultant mapping is therefore

$$\begin{aligned} B(\theta) &= 2 + [X'(\theta) - X'(0)] \times \frac{-4}{\Delta X'} \\ &= 2 + \left[\frac{26.81 \times \theta}{1960 \times \frac{2\pi}{F_s} + \theta} + 0.53 - 0.53 \right] \times \frac{-4}{\Delta X'} \\ &= 2 - \frac{\frac{26.81 \times 4}{\Delta X'} \times \theta}{1960 \times \frac{2\pi}{F_s} + \theta} \\ &= 2 - \frac{C_m \times \theta}{C_a + \theta} \end{aligned} \tag{22}$$

where

$$C_m = \frac{26.81 \times 4}{\Delta X'}$$

and

$$C_a = 1960 \times \frac{2\pi}{F_s}$$

Appendix D

The Tutor's Instruction Manual

THE TUTOR

A SPEECH THERAPY TUTOR FOR VOWEL PRODUCTION

JAMIE TURNBULL

September 1991

Dundee Institute of Technology in collaboration with the
Department of Medical Physics, Ninewells Hospital, Dundee

Important Note

To prevent damage to the Tutor it is essential that the following procedure must be taken before moving the Tutor to another location. Even moving the computer from one side of the desk to another could damage the machine if the proper steps are not taken.

To prepare some of the internal workings of the computer for transport the Tutor program should be terminated properly. This can be achieved by moving the cursor to the *Quit* option on the main menu and pressing the *Enter* key. The screen will clear and a further list of options will be displayed. Select option 3, "PREPARE SYSTEM FOR MOVING", and press *Enter*. The computer is now ready to be switched off and relocated.

It is always advisable to carry out these steps before switching off the machine, even if you do not intend to move it. If you are going to move the machine after someone else has used it and you are not sure whether the above procedure has been followed, switch on the machine and carry out the necessary steps.

Contents

List of Figures

1 Introduction

This Speech Therapy Tutor, henceforth referred to as "the Tutor", is based around an IBM PC compatible, and has been developed by the Speech and Signal Processing Group at Dundee Institute of Technology in collaboration with the Medical Physics Department at Ninewells Hospital.

The principal use of the Tutor should be in the treatment of patients with vowel production difficulties. It can handle vowel sounds in three different contexts, namely

- isolated vowels (V)
- consonant-vowel (CV) utterances
- consonant-vowel-consonant (CVC) utterances

In each of the above three cases the vowel sound must be extended. This allows the Tutor to detect automatically the region of the utterance which corresponds to the vowel sound alone.

The Tutor operates by comparing previously defined templates (see Section 3) with the results of analysing the patient's utterance. These templates are stored on a computer disk file called a database.

Each database contains up to ten entries, and each entry contains the results of analysing a particular sound. Included with each entry is some other useful information such as date of analysis, name, age, and sex of patient, and some general comments.

Two methods of feedback to the patient are available. These are known as the *deferred feedback mode*, and the *immediate feedback mode.*

In the deferred feedback mode, the whole utterance is analysed before any comparison is made. Hence the feedback is deferred until the end of the utterance. In the immediate feedback mode, results of the analysis are displayed to the patient as the sound is being produced.

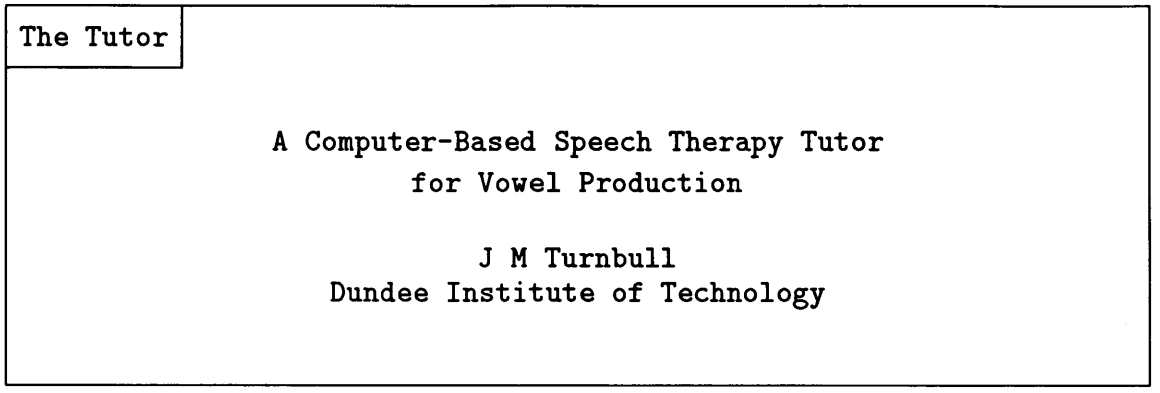

Figure 1: The introductory screen that appears when the Tutor is switched on.

2 Getting Started

Plug the computer into a free mains socket, and switch it on. The computer will whirr, bleep and make some rattling noises for about thirty seconds.

If all goes well, the screen shown in Figure 1 will appear.

This introductory screen will persist until a key on the keyboard is pressed. When you press any key to continue, you will be taken directly to the *Select database* option, which is described in the following section.

3 Selecting a Database File

The next screen after the first one is a list of all database files which are currently stored on disk. It will look similar to the screen shown in Figure 2.

You will notice that on the computer screen the first entry is highlighted. A database is chosen by moving the highlighted region, called a cursor, to the desired entry in the list and pressing the ENTER key.

The cursor is moved up and down, and from left to right by pressing the corresponding arrowed keys on the keypad which is situated to the right of the conventional QWERTY keyboard.

Once a database has been selected, the computer will load the information from

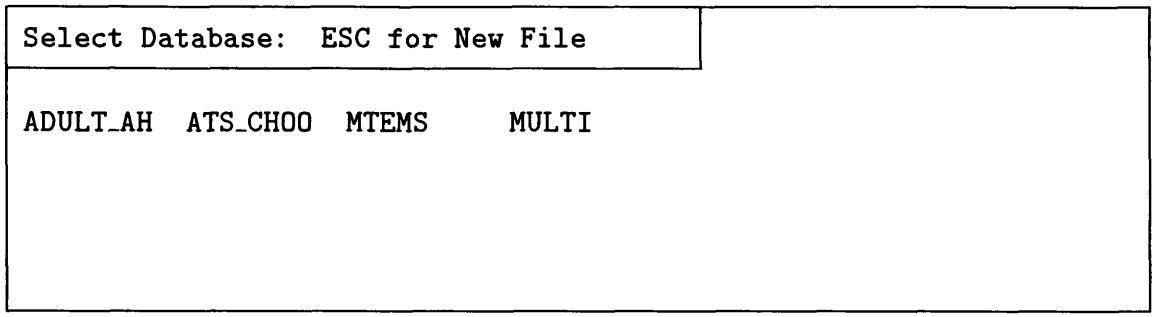

Figure 2: The screen shown when selecting a database.

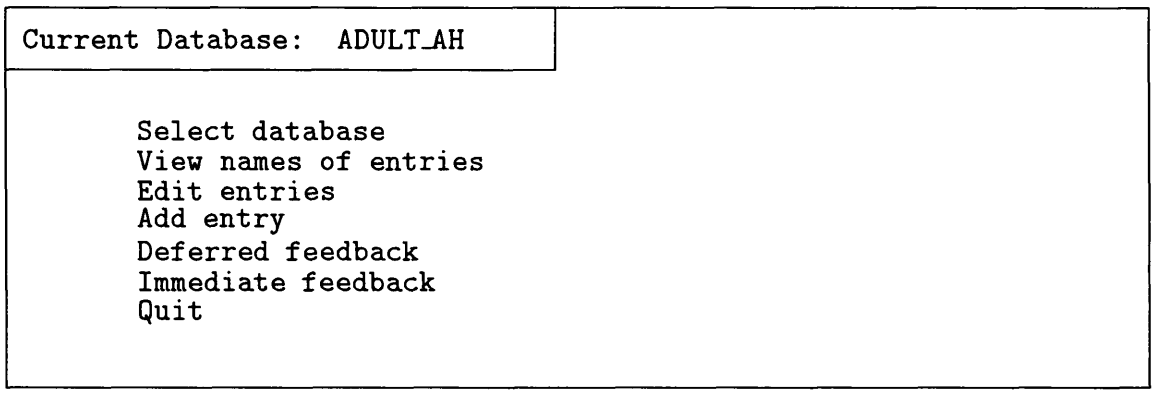

Figure 3: The main menu screen.

disk and proceed to the main menu screen.

4 The Main Menu Screen

The main menu screen will resemble the screen shown in Figure 3. It is displayed once a database has been chosen after start-up, and will also be displayed after each command or function available from this menu has been completed. Notice that the currently selected database is shown in the top left of the screen.

In a manner similar to that in the *Select database* screen, a cursor highlights the command or option that will be invoked when the ENTER key is pressed. The cursor

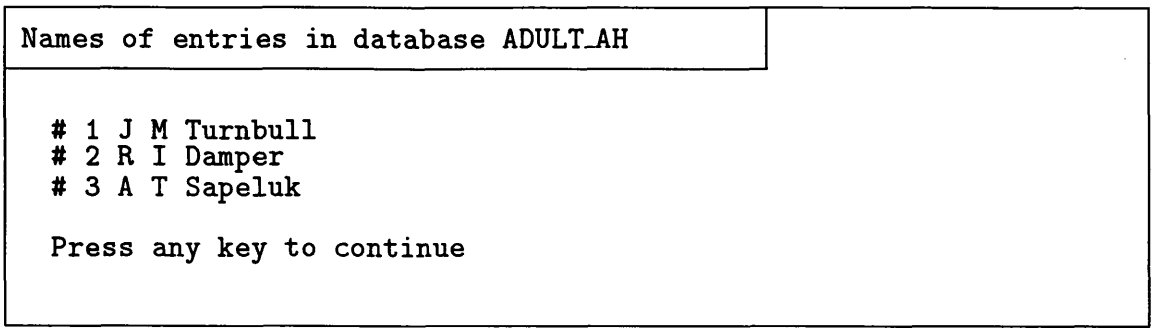

Figure 4: Viewing the names of all entries in the current database.

is moved using the up and down arrow keys. Also, the cursor can be moved directly to an option by pressing the key which is the first letter of that option, e.g. if you press "E" or "e" then the *Edit entries* option will be highlighted.

When this menu is shown for the first time, the cursor will highlight the *Select database* option. Pressing enter at this point will take you back to the database selection screen (see Section 3).

The main menu screen also offers an on-line help facility. To obtain on-line information about a particular main menu option, move the cursor to that option and then press the *F1* key. A short explanatory note will be displayed. Once you have finished reading the help message you can return to the main menu by pressing any key.

In order to explain the rest of the functions available, it will be assumed that the database `ADULT_AH` has been loaded using the *Select database* function.

4.1 View Names of Entries

This option allows the therapist to see at a glance the names of the patients who have been used to create a particular database. For example, if the database `ADULT_AH` has been loaded, this function will show a display not unlike the screen shown in Figure 4. Once you have finished viewing the names pressing any key will return you to the main menu.

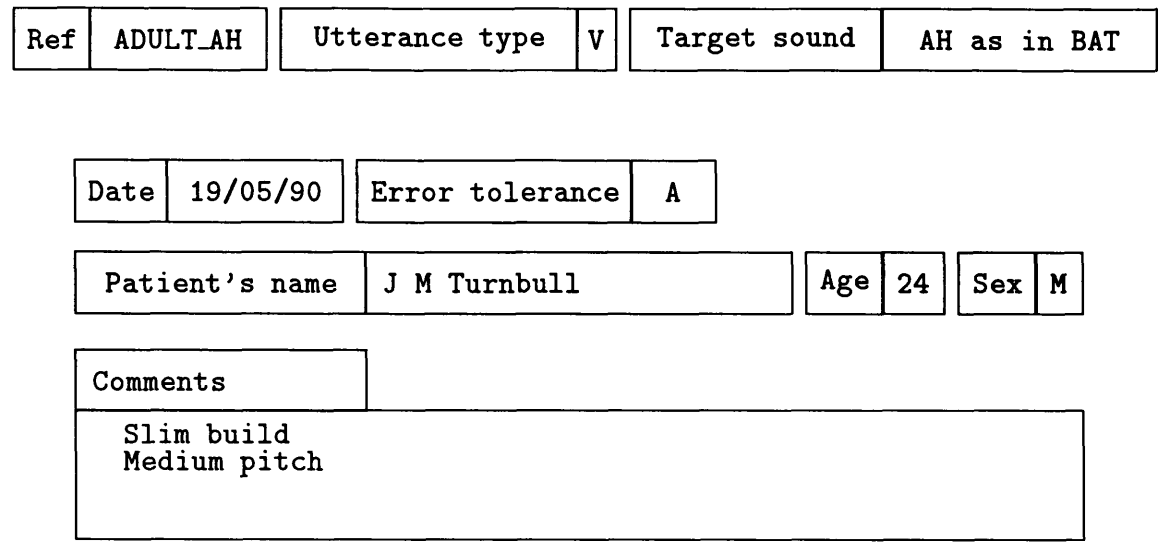

Figure 5: Editing a database entry.

4.2 Edit Entries

This option can be used to edit an existing database entry or simply to view the contents of an entry. When this option is selected, the first entry in the database will be displayed together with information about the database as a whole. It will look something like Figure 5.

The top three boxes give the information about the database as a whole and is supplied by the user when a new database is created (see Section 5). The contents of these boxes cannot be altered.

The first of these three boxes, labeled "Ref", gives the name of the current database. The second, "Utterance Type", describes the type of utterance in which the vowel sound was detected. For example, this can be V for isolated vowels, CV for vowel sounds in a consonant-vowel context, and CVC for vowels in a consonant-vowel-consonant context. The third box, "Target Sound", gives a brief description of the vowel sound of interest.

The next six boxes give the information for each entry in the database. The first of these, "Date", shows the date that the entry was added to the database. The

next box, "Error Tolerance", gives an indication of how well the patient can perform. This value should range from "A" (for an excellent utterance) to "F" (for a very poor utterance).

The third box, "Patient's Name", gives the identity of the patient. This box need not necessarily contain the patient's name, but could contain a number or code which tells the therapist which patient produced the utterance. The next two boxes contain the patient's age and sex ("M" or "F").

The final box gives three lines of text in which the therapist can make any comments or add any other relevant information.

When you have just selected the *Edit entry* function you will notice that the contents of the "Date" box are highlighted. The highlighting indicates which line you are currently editing. It can be moved by pressing the *up-arrow* key, the *down-arrow* key, or the *Enter* key. (Experiment by pressing these keys to move the highlighted region from box to box).

Note that the *Num Lock* light on the keyboard must not be lit. It can be switched on and off by pressing the *Num Lock* key just below the light on the keyboard.

A second thing you will notice is the flashing bar which sits below a letter in the highlighted region. This bar is called a *text cursor*. It shows the position in the line that characters will be inserted. If you press any of the numeric or alphabetic character keys the corresponding character will be inserted after the character to the left of the cursor (provided there is some blank space at the end of the line).

Characters to the left of the cursor can be deleted by pressing the dark grey left arrow key next to # key. The character under which the cursor is flashing can be deleted by pressing the *Del* key. The whole line can be deleted by pressing the *Ctrl* key and the dark grey left arrow key simultaneously.

When you have finished viewing or altering that entry, you can proceed directly to the next entry by pressing the *PgDn* key, or return to the main menu by pressing the *Esc* key. You can go back to the previous entry by pressing the *PgUp* key. Pressing *PgUp* at the first entry will have no effect, and pressing *PgDn* at the last entry will return you to the main menu.

As with the main menu screen, there is an on-line help facility built into the *Edit*

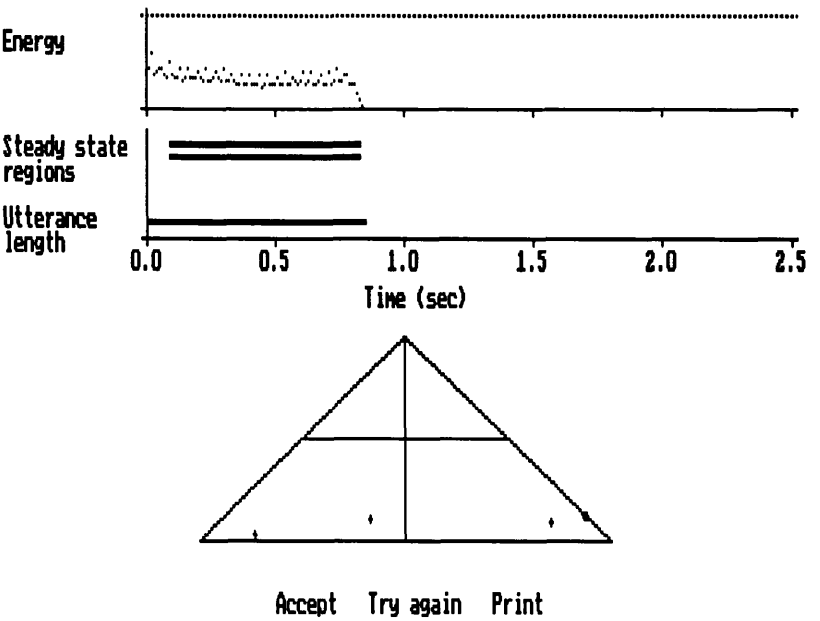

Figure 6: A typical display shown after analysing a monosyllabic word in the *Add entry* option. This example shows a single un-fragmented steady state region.

entries option. Pressing the *F1* key to the left of the keyboard will replace the current screen with some information about the current box. For example if you are currently editing the contents of the "Date" box, then pressing *F1* will give some information about that box. Any further key-press will return you to the edit screen.

4.3 Add Entry

To add an entry to the database, this option is selected. Note that there is a maximum limit of ten entries per database. This option comprises two sections. Firstly, monosyllabic utterances will be analysed until the therapist is satisfied that the last utterance analysed was a good example. Once the utterance has been accepted the results will be stored and supplementary information (such as patient's name, age, etc.) must be entered by the therapist.

During the analysis stage, two pairs of axes and a triangle will be drawn (see Figure 6) and the computer will bleep when it is ready to analyse a sound. For an explanation of what the triangle represents, see Section 6. The top trace is a plot

of the energy or loudness of the sound against time. As a sound is made into the microphone the energy plot will be drawn. The maximum duration of the utterance is just over 2.5 seconds, so this trace will also show you when you are getting near the maximum duration.

When the sound is completed, the utterance will be analysed to determine the steady state regions of the sound. These regions will occur when the articulators in the vocal tract are stationary. If no steady state sounds are detected, a message will be displayed asking for the utterance to be repeated.

If steady state regions are detected, some circles will be drawn on the triangle (see Section 6). Also drawn will be some lines on the trace below the energy plot.

The lowest and longest line shows the length of the utterance. The lines above this show the steady state regions of the utterance. In most cases where a patient is capable of producing a continuous sound these lines will look similar to the ones shown in Figure 6.

On occasions the Tutor may detect more than one steady state region: this may happen, for example, if the patient's voice falters during the production of the vowel. On these occasions the trace will more likely look like the one shown in Figure 7. You will notice that the longest steady state region is displayed as a double bar. This is the region that is used to generate the template.

This display of the steady state regions may be useful in training a patient to produce a continuous sound. However, it would be more useful to use the *Deferred feedback* mode for this purpose (see Section 4.4) as you would not need to enter any further database details.

You will notice the words "Accept", "Try again", and "Print" at the bottom of the screen. If the longest steady state region is of an acceptable length (say greater than one third that of the utterance) and the utterance was a good example then the corresponding section of the utterance can be accepted as the template by pressing "A" or "a", and subsequently you will be asked to enter the patients details. If you feel that the utterance was not a good example then you can press "T" or "t" to try again.

If you have a printer connected to the Tutor, you can print what is shown on the

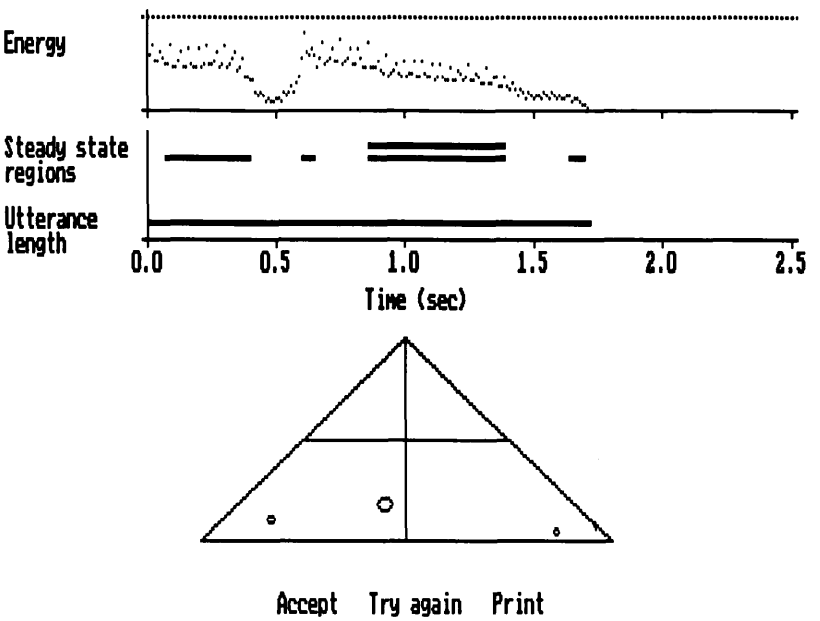

Figure 7: A typical display shown after analysing a monosyllabic word in the *Add entry* option which shows several steady state regions detected.

screen by pressing "P" or "p".

4.4 Deferred Feedback

The *Deferred feedback* mode is very similar to the *Add entry* option, except that there is an additional semi-circle on the display (see Figure 8). This displays a measure of the match between the utterance and the nearest (or best match) pre-defined template.

The method of display uses a dial which moves clockwise around the display. When the dial is at the extreme left then the utterance is deemed to be markedly different from the template, hence it is a poor quality reproduction of the template sound (it could however be a good quality reproduction of an other sound). When the dial is at the extreme right it is a very close match to the template.

The number that is shown at the *point* of the dial corresponds to the template which the utterance was closest to.

You will notice that the words "Menu", "Try again", and "Print" are displayed

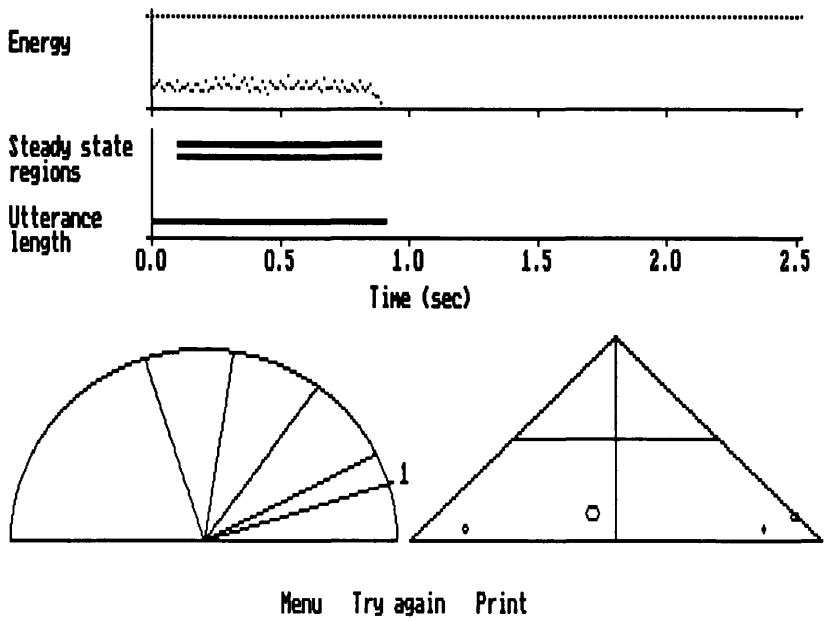

Figure 8: Display shown after analysing a monosyllabic word using the *Deferred feedback* option.

at the bottom of the screen. If "M" is pressed you are returned to the main menu. If a "T" is pressed the computer will clear the screen and bleep, indicating that it is ready to analyse more speech.

If you have a printer connected to the Tutor, you can print what is shown on the screen by pressing "P".

4.5 Immediate Feedback

The *Immediate feedback* mode is different from the *Add entry* and *Deferred feedback* modes. In this case the results of the analysis are displayed as the utterance is being produced.

With this mode, patients can experiment with moving their tongue, jaw, and lips, while watching the screen to see immediately if the sound is close to or markedly different from the target sound. This allows them to learn the correct vocal tract shape for a particular sound by experimenting and observing the results.

The method of display uses four flags moving up and down flagpoles which are

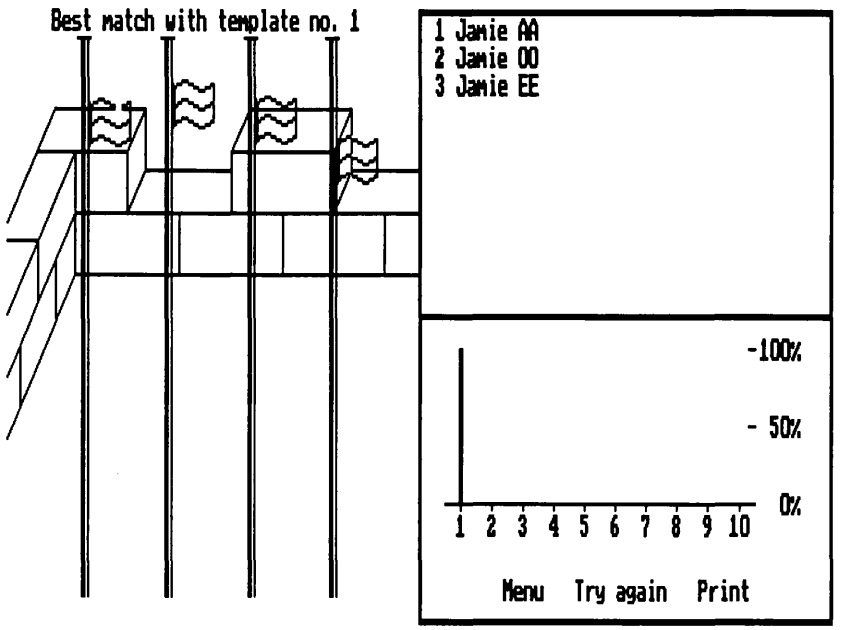

Figure 9: Display shown after analysing an utterance in the *Immediate feedback* mode.

situated on the top of a castle tower (see Figure 9). As the sound is being produced, it is analysed and matched against *all* templates in the current database. The heights of the flags correspond to the closeness of the match, i.e. the higher the flags the closer the utterance is to that of the nearest template.

When the sound is stopped, the Tutor will list the names of all the current database entries, and display a histogram of the number of matches to each template. For example, in Figure 9 it is seen that the utterance matched with the first template for 100% of the time.

Again the words "Menu", "Try again", and "Print" are displayed at the bottom of the right half of the screen. If an "M" is pressed you are returned to the main menu. If a "T" is pressed the computer will clear the screen and bleep, indicating that it is ready to analyse more speech.

If you have a printer connected to the Tutor, you can print what is shown on the screen by pressing "P".

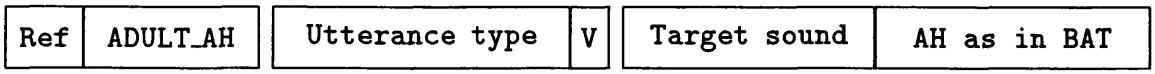

Figure 10: A typical screen-shot from the *Select database* option.

4.6 Quit

To prevent damage to the Tutor it is essential that the following procedure must be taken before moving the Tutor to another location. Even moving the computer from one side of the desk to another could damage the machine if the proper steps are not taken.

To prepare some of the internal workings of the computer for transport the Tutor program should be terminated properly. This can be achieved by moving the cursor to the *Quit* option on the main menu and pressing the *Enter* key. The screen will clear and a further list of options will be displayed. Select option 3, "PREPARE SYSTEM FOR MOVING", and press *Enter*. The computer is now ready to be switched off and relocated.

It is always advisable to carry out these steps before switching off the machine, even if you do not intend to move it. If you are going to move the machine after someone else has used it and you are not sure whether the above procedure has been followed, switch on the machine and carry out the necessary steps.

5 Creating a New Database

To create a new database, return to the main menu and choose the *Select database* option. When the list of files currently on disk is displayed, press the ESC key.

The boxes shown in Figure 10 will be displayed, and the area in the first one highlighted. Moving around the boxes is achieved in the same way as in the *Edit entries* option, i.e. use the *Up* and *Down* arrow keys to move between boxes, and the *Left* and *Right* arrow keys to move the text cursor within the box. Characters can also be deleted and inserted as before (see Section 4.2).

There is an on-line help facility provided with this option. Pressing the *F1* key in this mode will give a help message for the box that you are currently working with.

In the first box you must provide the computer with the name of the new database that will be created. This name must have a maximum of eight characters and comprise only letters, numbers, and the underscore (_).

In the second box, type in the type of utterance that this database will correspond to, i.e. V, CV or CVC. This is for information purposes only: it will tell future users of that database file the purpose for which it was intended.

In the third box, type in a short description of the vowel sound that the database will represent. This is also for information purposes only.

When you are happy with the contents of all boxes, press the ESC key to return to the main menu.

6 Technical Information

For voiced speech, the sound originates at the vocal folds. The articulators in the vocal tract produce constrictions and cavities that selectively enhance certain frequency components of the sound called the *formant frequencies.*

These formant frequencies appear as peaks in the frequency spectrum of the sound produced. Some peaks are narrow and some are broad. The width of the peak is called the *bandwidth.*

The triangular display that is shown in the *Add entry* and *Deferred feedback* options shows the positions of these frequencies and a measure of the bandwidths as shown in Figure 11. The diagram shown is actually greatly simplified but should suffice for our purposes.

Figure 11 shows the position of four formant frequencies as small circles. You will notice that the radii of the circles can vary. This actually shows the standard deviation or movement of the formant about its mean position.

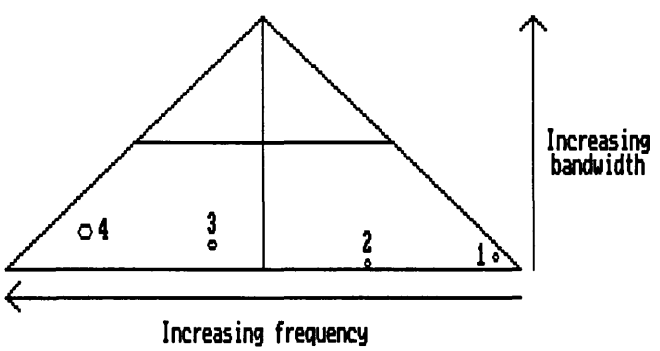

Figure 11: The triangular display shown in the *Add entry* and *Deferred feedback* mode gives a measure of the frequency and bandwidth of a formant.

Appendix E

Figures for LP Model Order Experiment

This appendix contains all of the results obtained from the experiment described in Chapter 5. Figures 32 to 63 are for the low noise experiment. The remaining figures are for the added noise experiment.

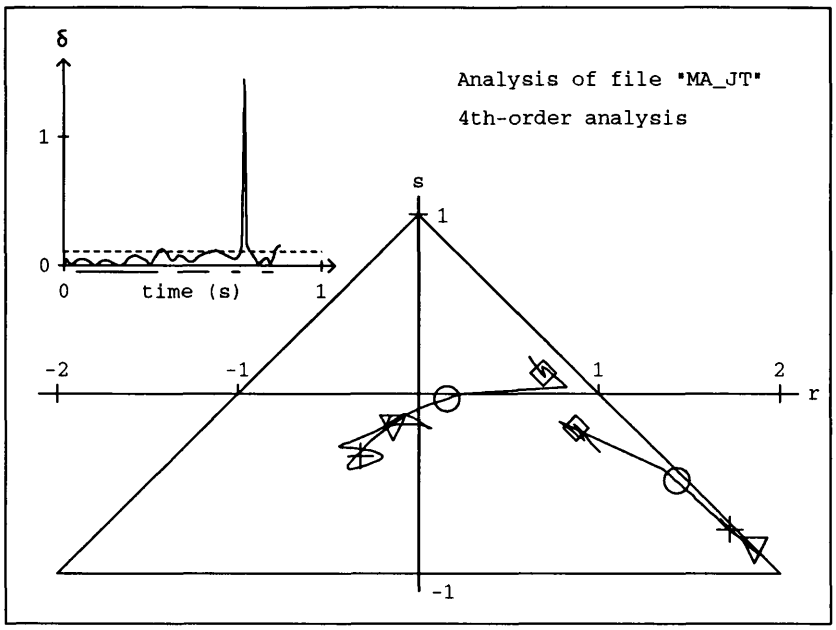

Figure 32: Analysis of /ma/ (speaker JT) using a 4th-order model.

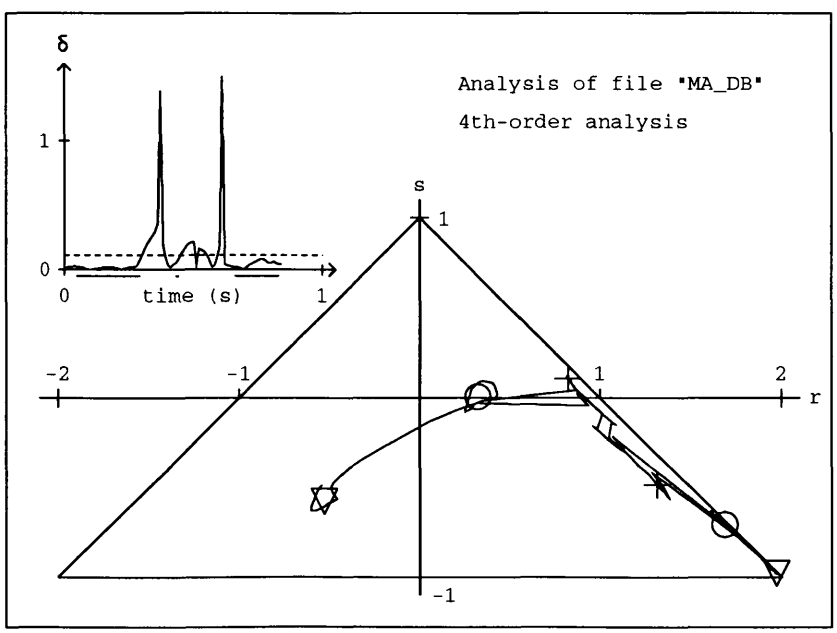

Figure 33: Analysis of /ma/ (speaker DB) using a 4th-order model.

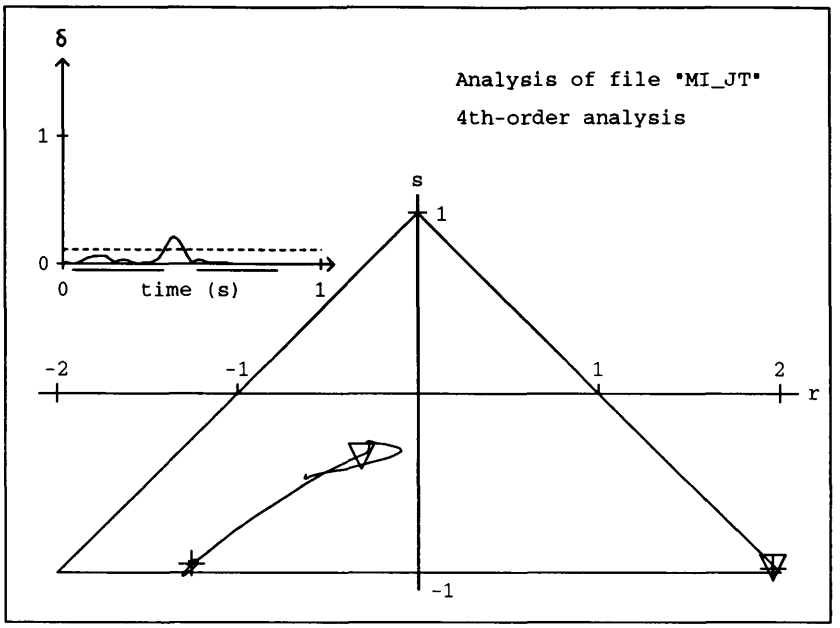

Figure 34: Analysis of /mi:/ (speaker JT) using a 4th-order model.

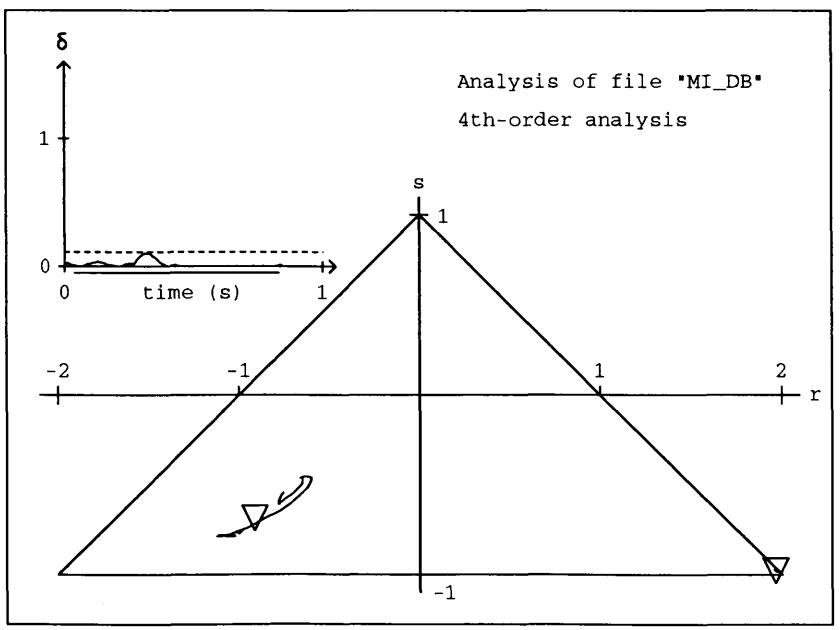

Figure 35: Analysis of /mi:/ (speaker DB) using a 4th-order model.

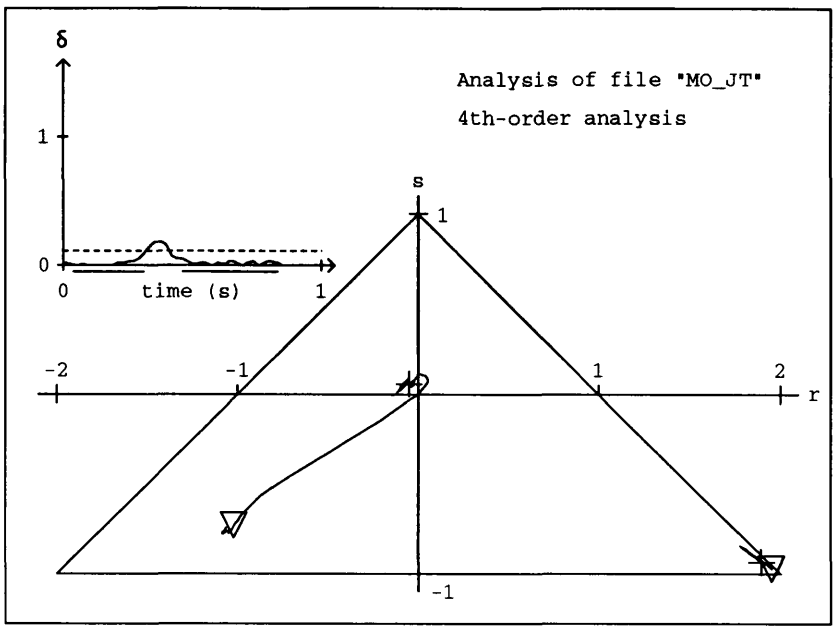

Figure 36: Analysis of /mo/ (speaker JT) using a 4th-order model.

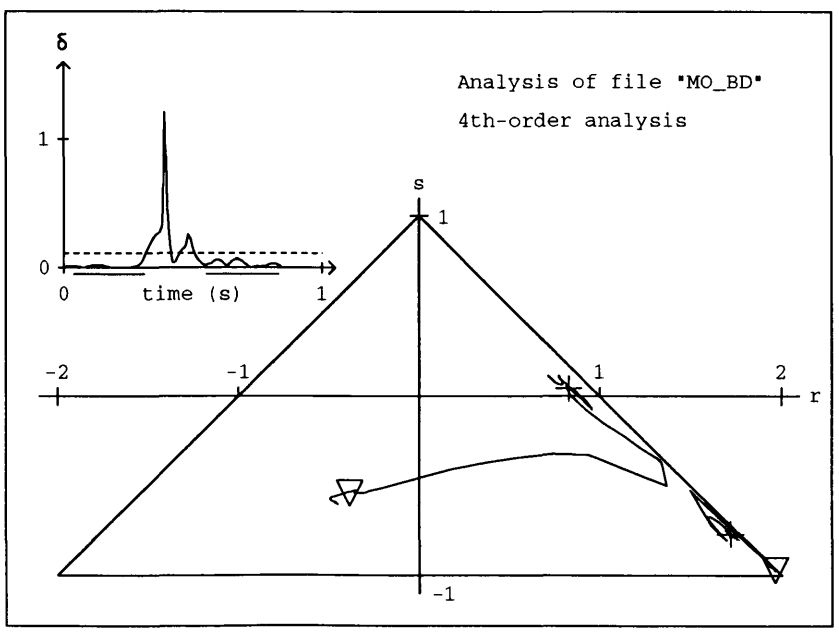

Figure 37: Analysis of /mo/ (speaker BD) using a 4th-order model.

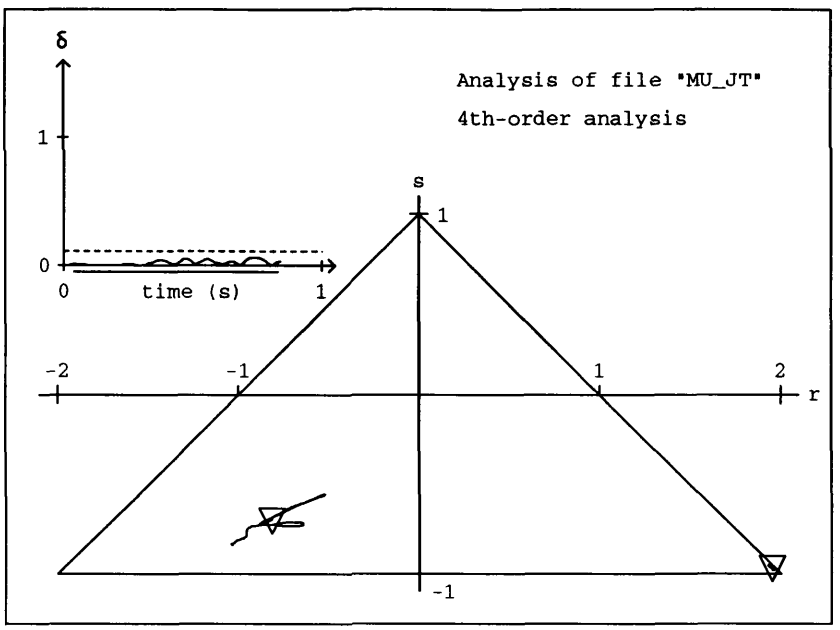

Figure 38: Analysis of /mu/ (speaker JT) using a 4th-order model.

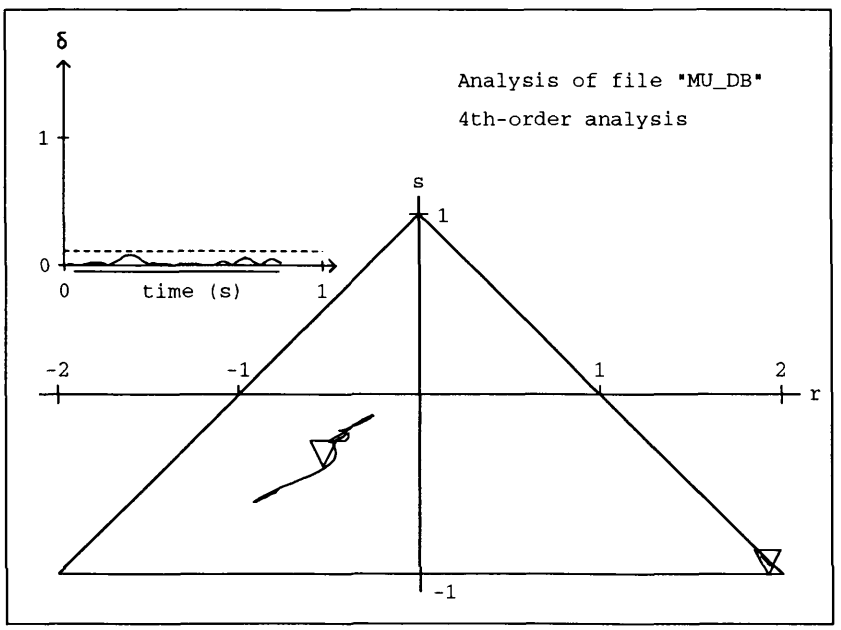

Figure 39: Analysis of /mu/ (speaker DB) using a 4th-order model.

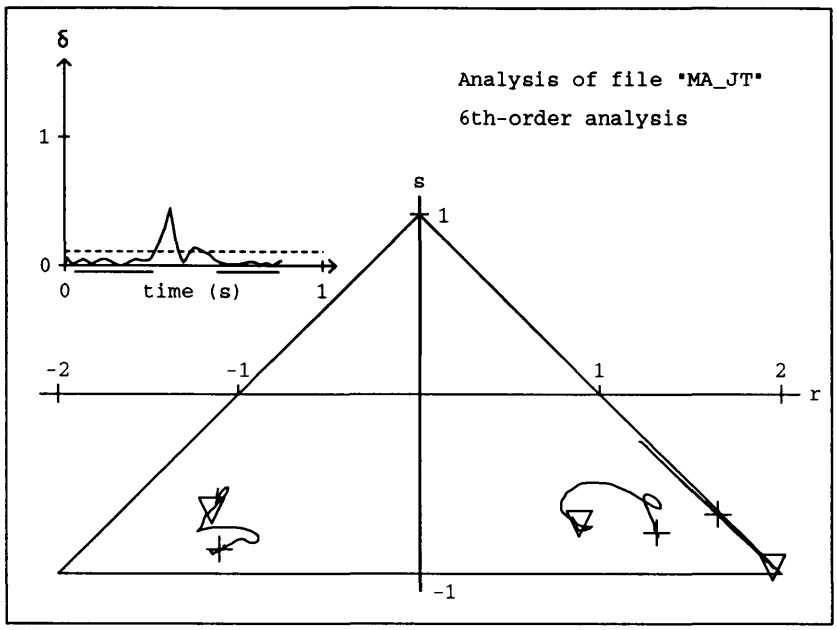

Figure 40: Analysis of /ma/ (speaker JT) using a 6th-order model.

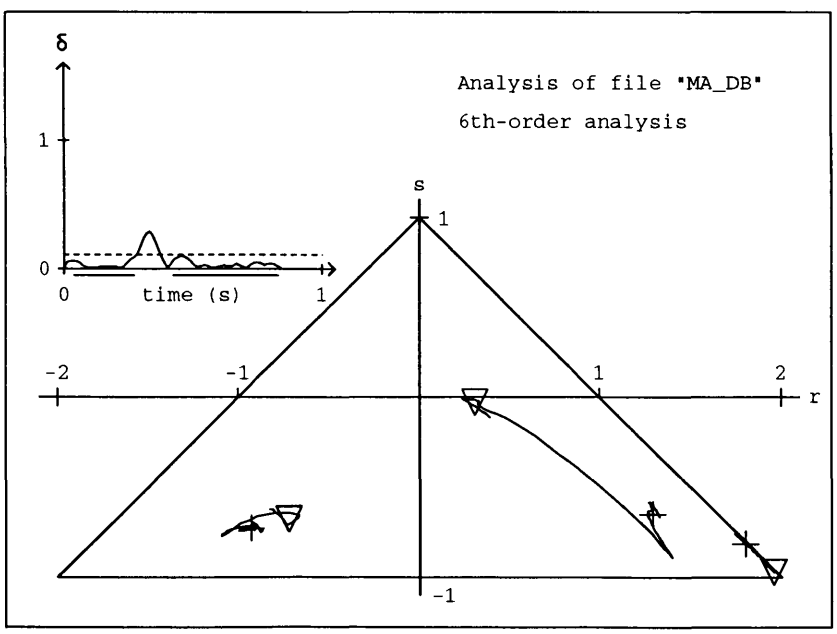

Figure 41: Analysis of /ma/ (speaker DB) using a 6th-order model.

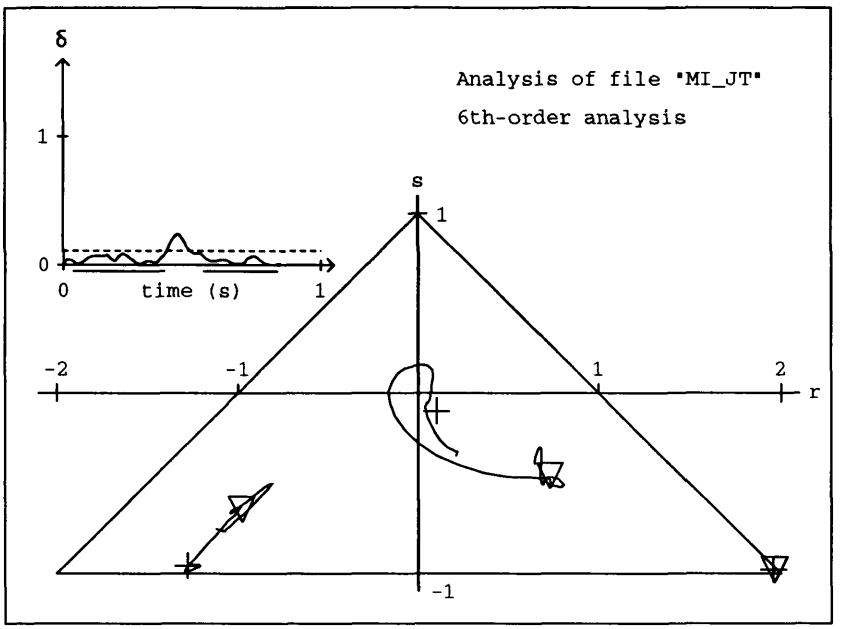

Figure 42: Analysis of /mi:/ (speaker JT) using a 6th-order model.

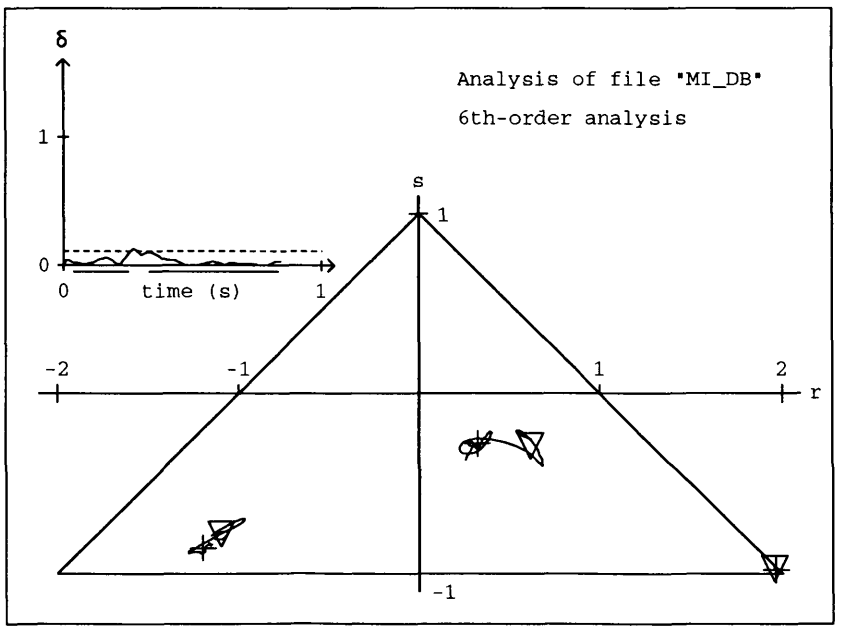

Figure 43: Analysis of /mi:/ (speaker DB) using a 6th-order model.

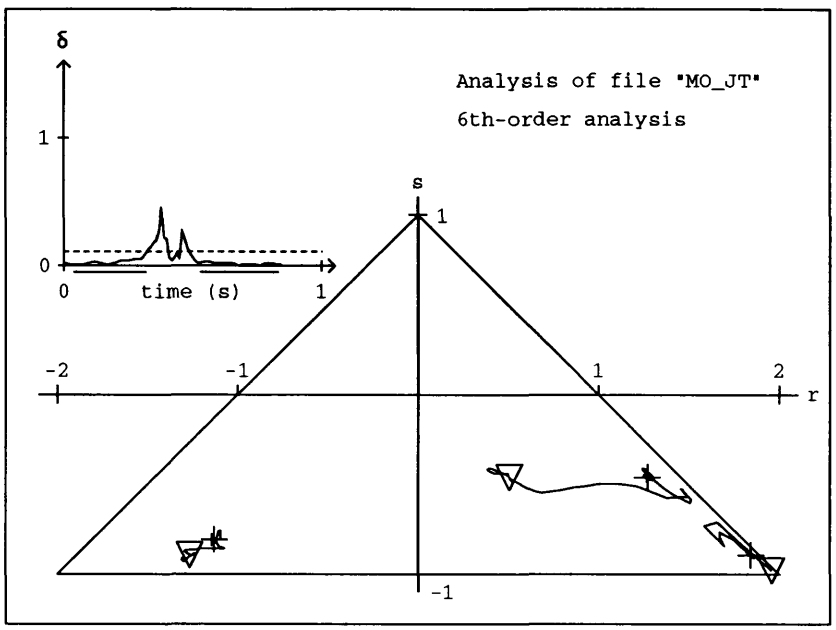

Figure 44: Analysis of /mo/ (speaker JT) using a 6th-order model.

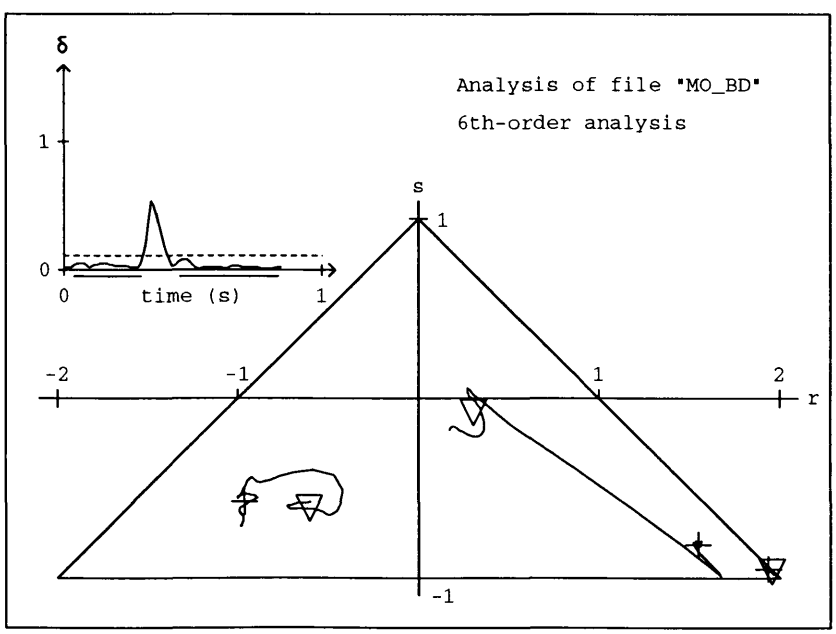

Figure 45: Analysis of /mo/ (speaker BD) using a 6th-order model.

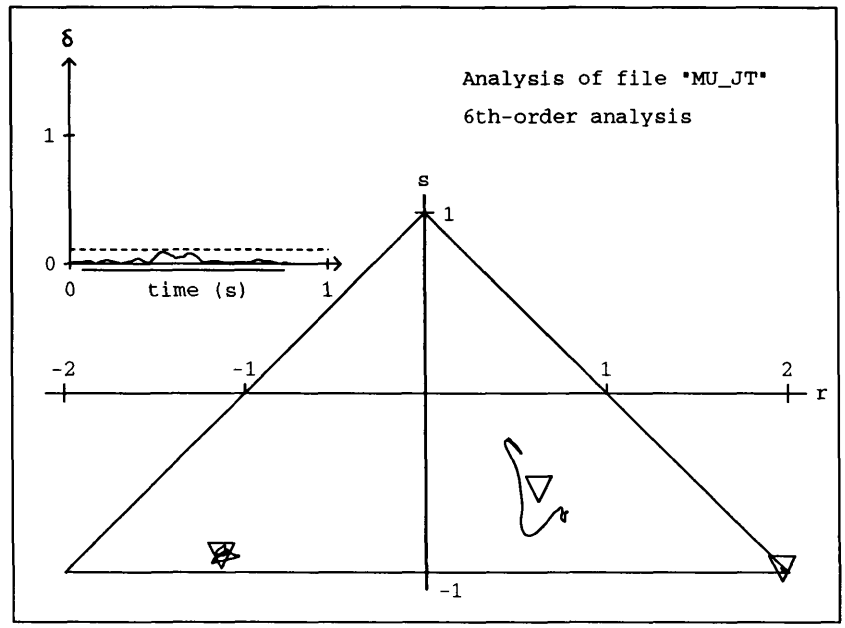

Figure 46: Analysis of /mu/ (speaker JT) using a 6th-order model.

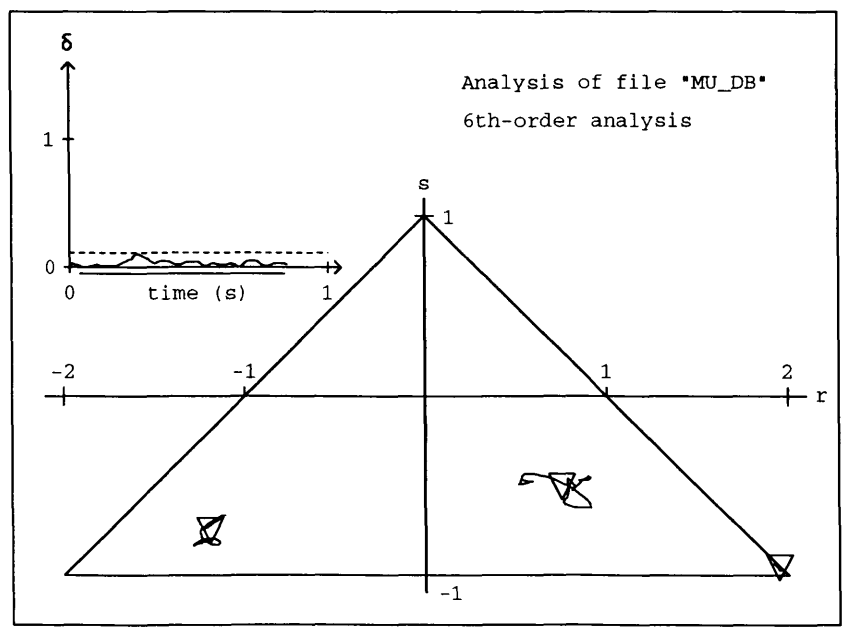

Figure 47: Analysis of /mu/ (speaker DB) using a 6th-order model.

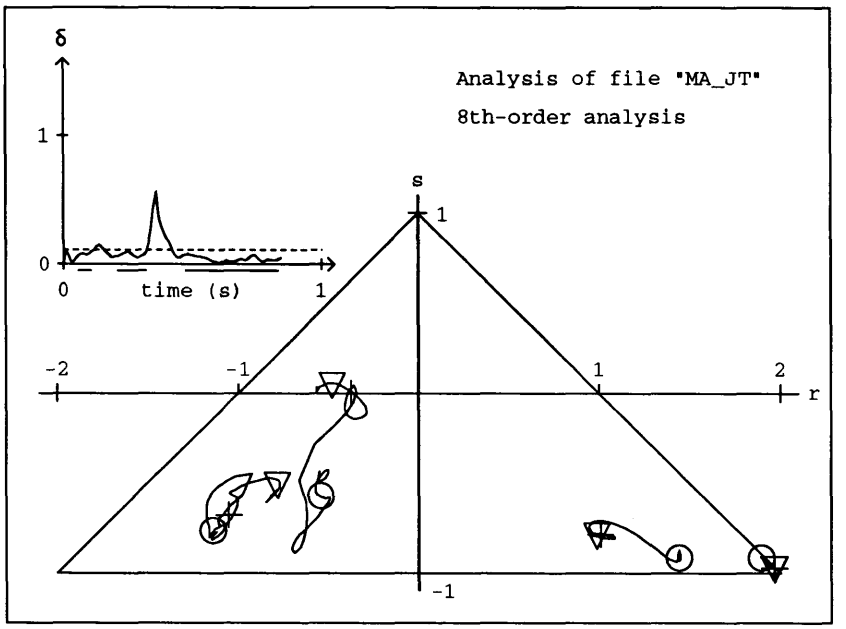

Figure 48: Analysis of /ma/ (speaker JT) using a 8th-order model.

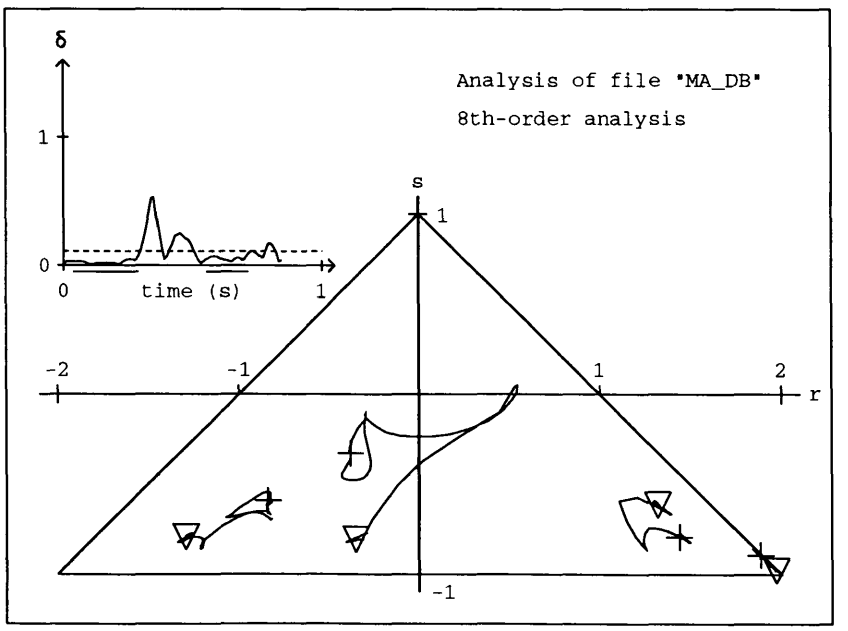

Figure 49: Analysis of /ma/ (speaker DB) using a 8th-order model.

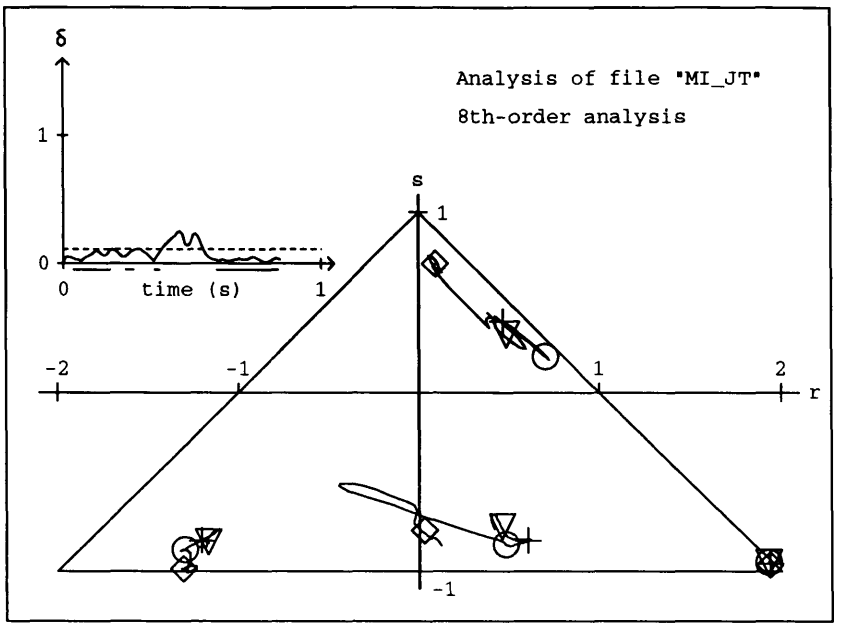

Figure 50: Analysis of /mi:/ (speaker JT) using a 8th-order model.

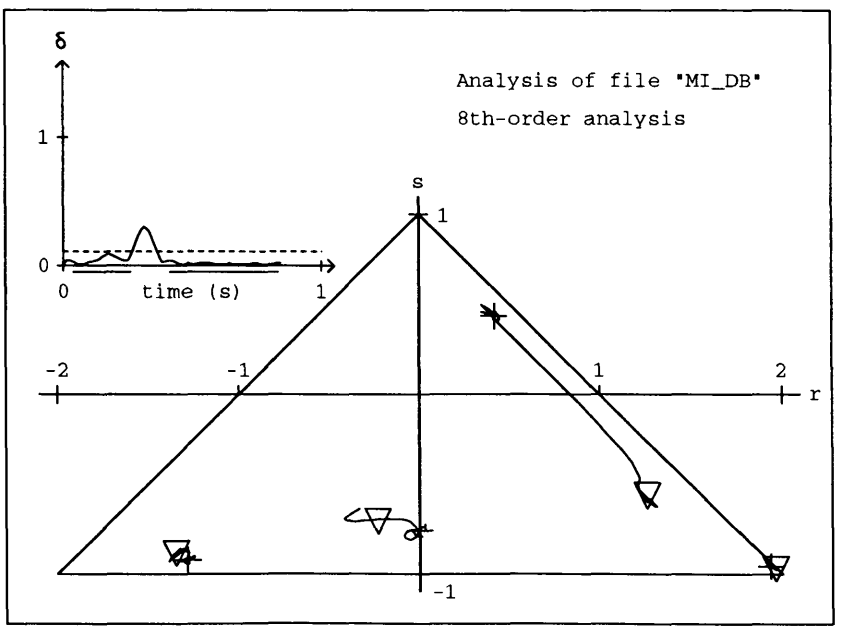

Figure 51: Analysis of /mi:/ (speaker DB) using a 8th-order model.

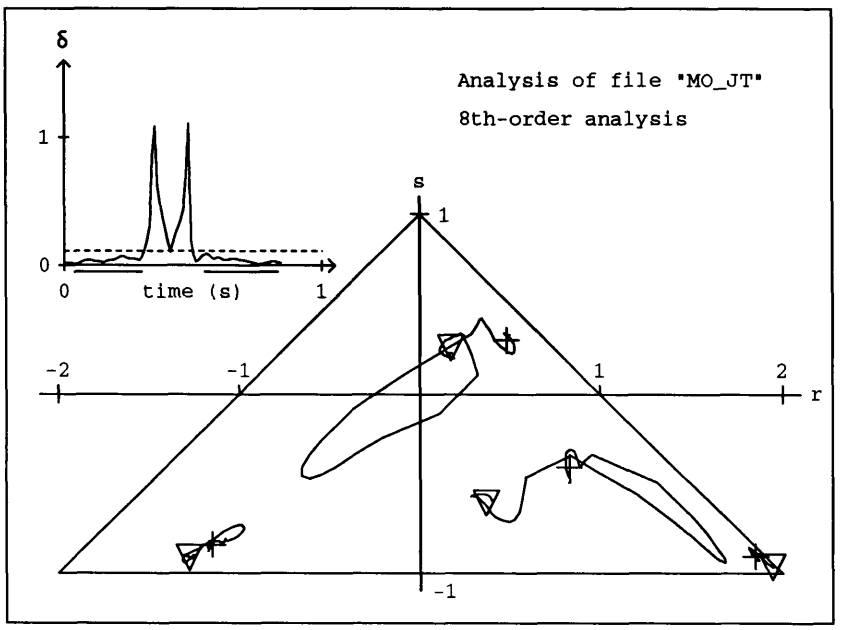

Figure 52: Analysis of /mo/ (speaker JT) using a 8th-order model.

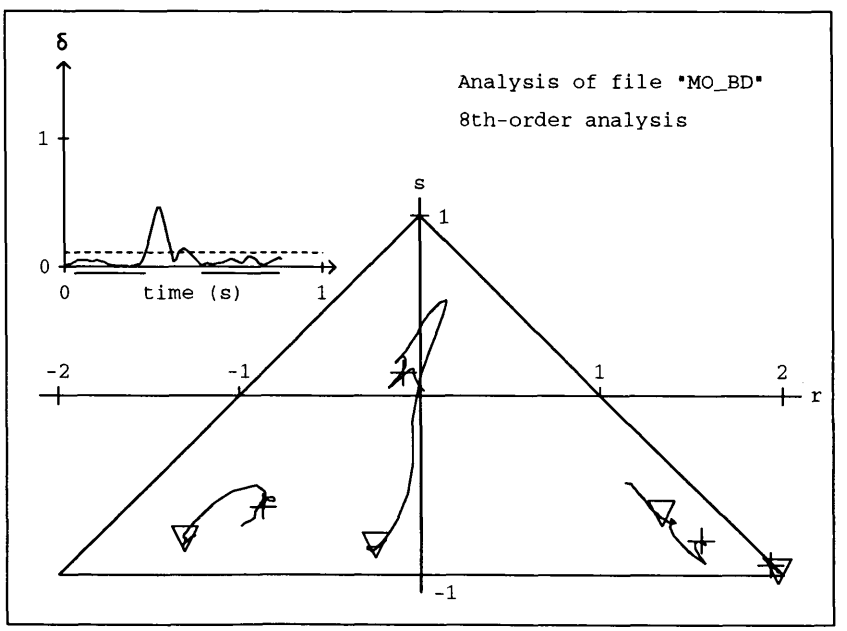

Figure 53: Analysis of /mo/ (speaker BD) using a 8th-order model.

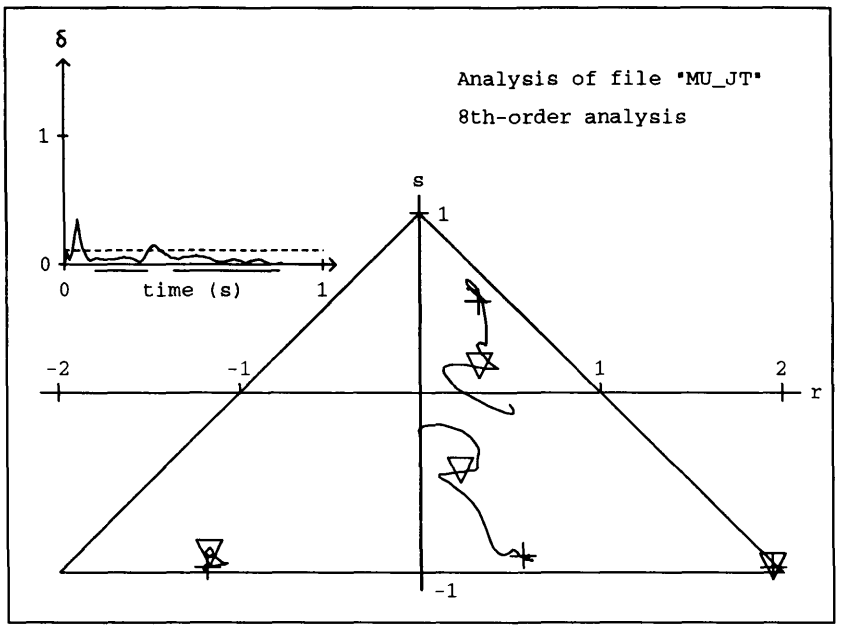

Figure 54: Analysis of /mu/ (speaker JT) using a 8th-order model.

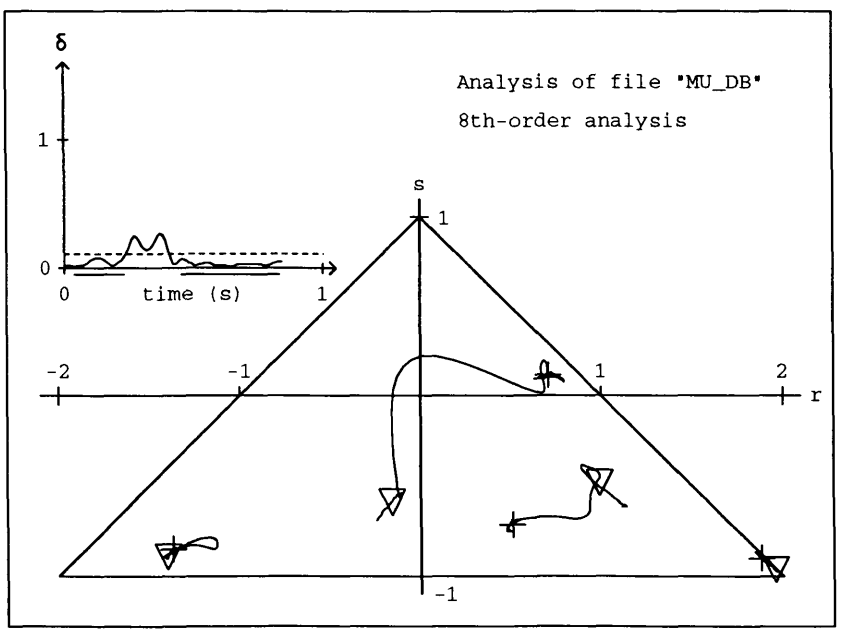

Figure 55: Analysis of /mu/ (speaker DB) using a 8th-order model.

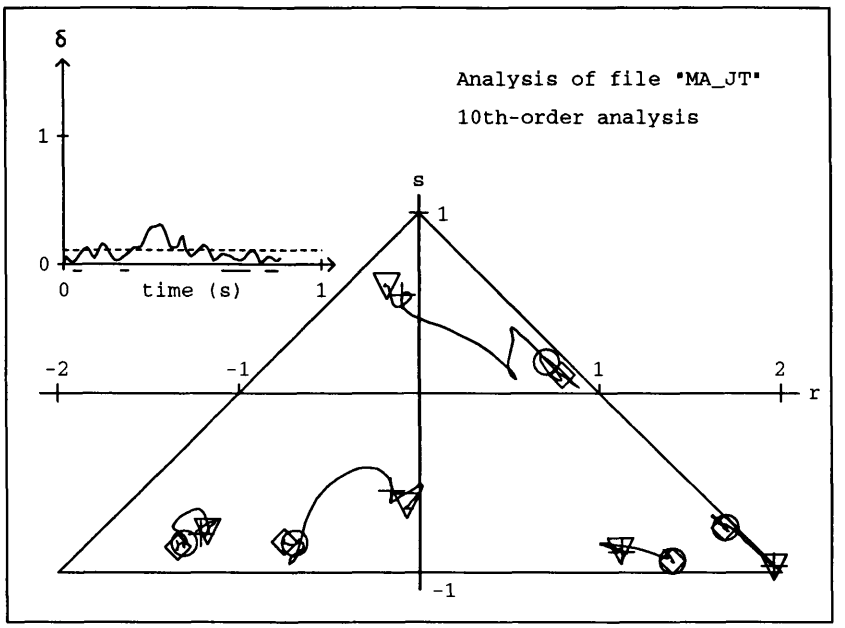

Figure 56: Analysis of /ma/ (speaker JT) using a 10th-order model.

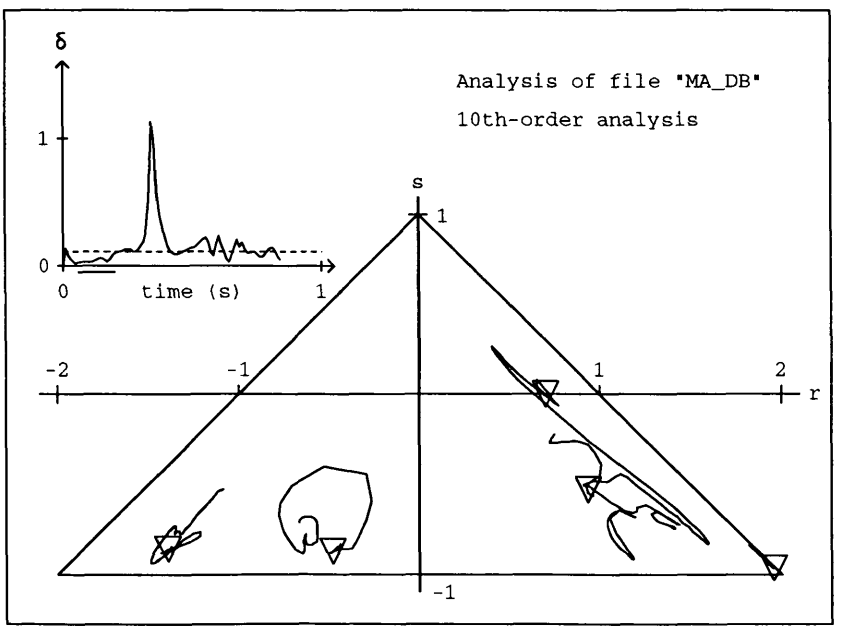

Figure 57: Analysis of /ma/ (speaker DB) using a 10th-order model.

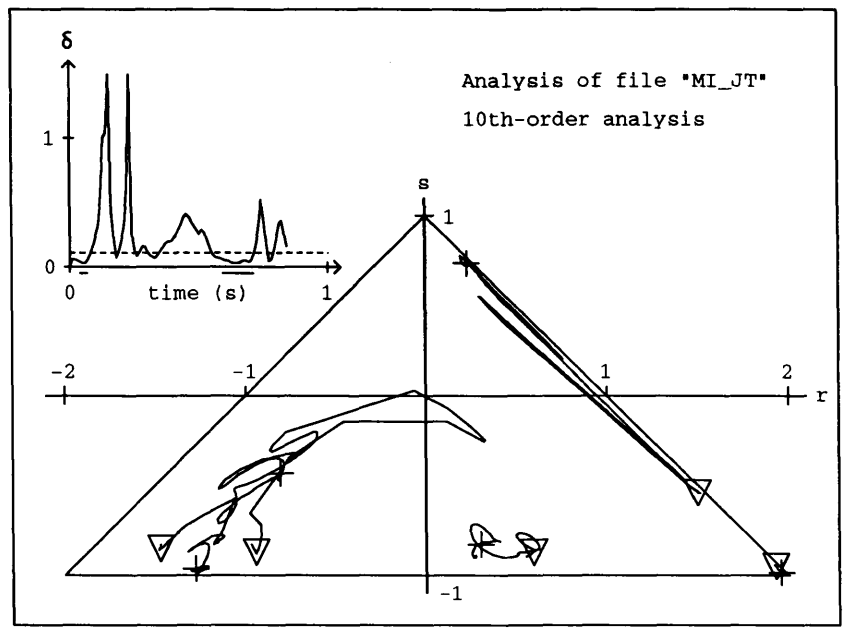

Figure 58: Analysis of /mi:/ (speaker JT) using a 10th-order model.

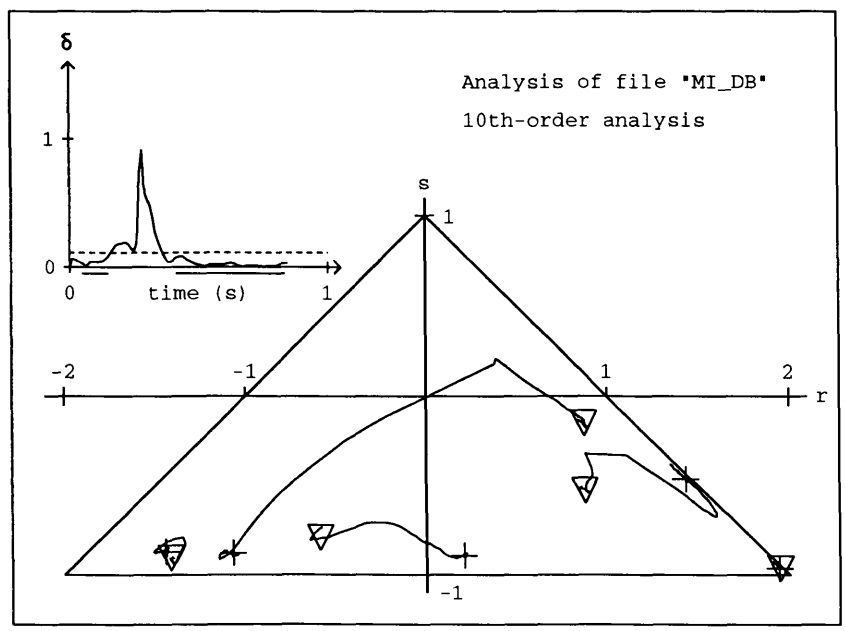

Figure 59: Analysis of /mi:/ (speaker DB) using a 10th-order model.

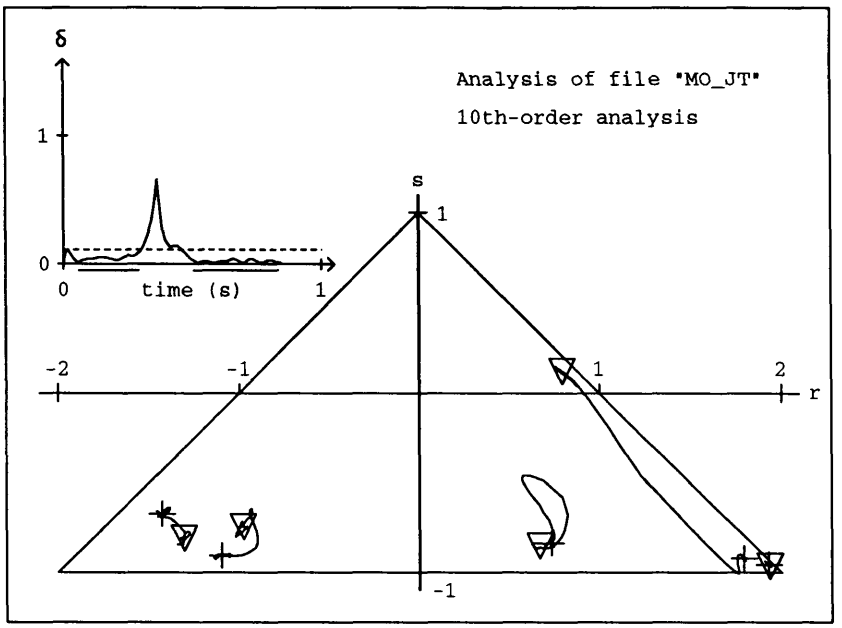

Figure 60: Analysis of /mo/ (speaker JT) using a 10th-order model.

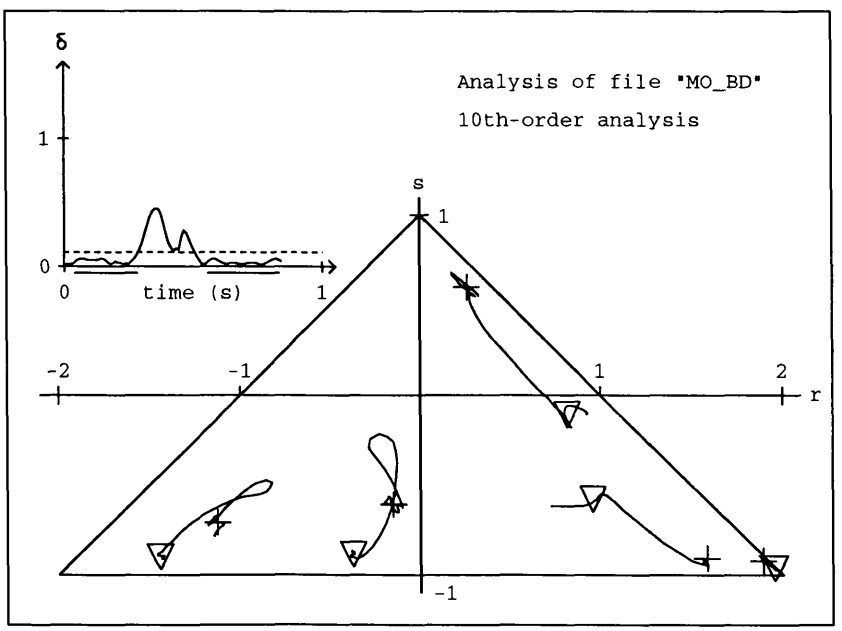

Figure 61: Analysis of /mo/ (speaker BD) using a 10th-order model.

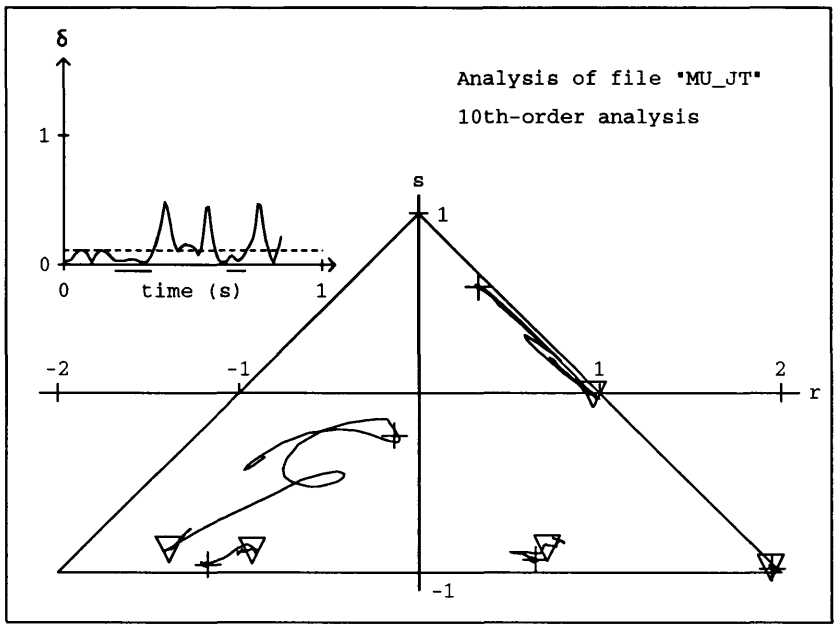

Figure 62: Analysis of /mu/ (speaker JT) using a 10th-order model.

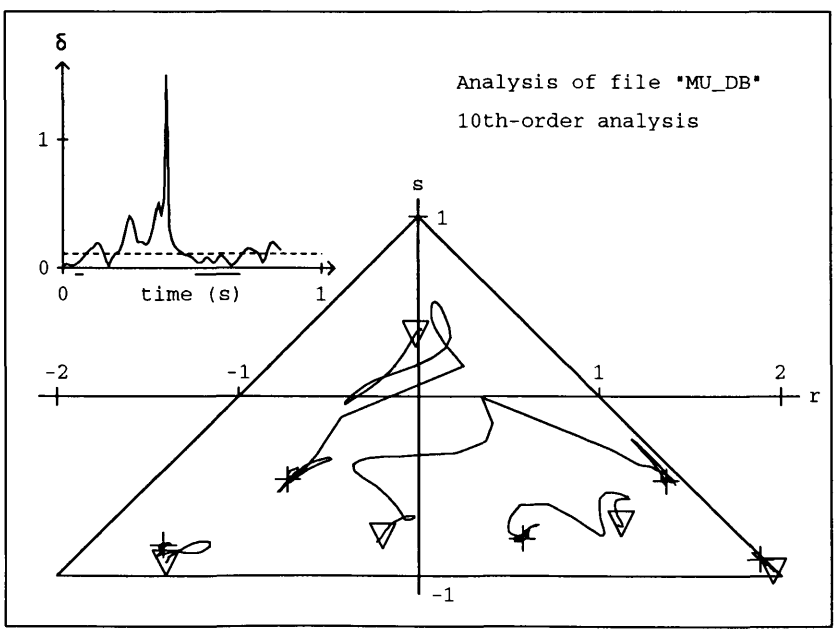

Figure 63: Analysis of /mu/ (speaker DB) using a 10th-order model.

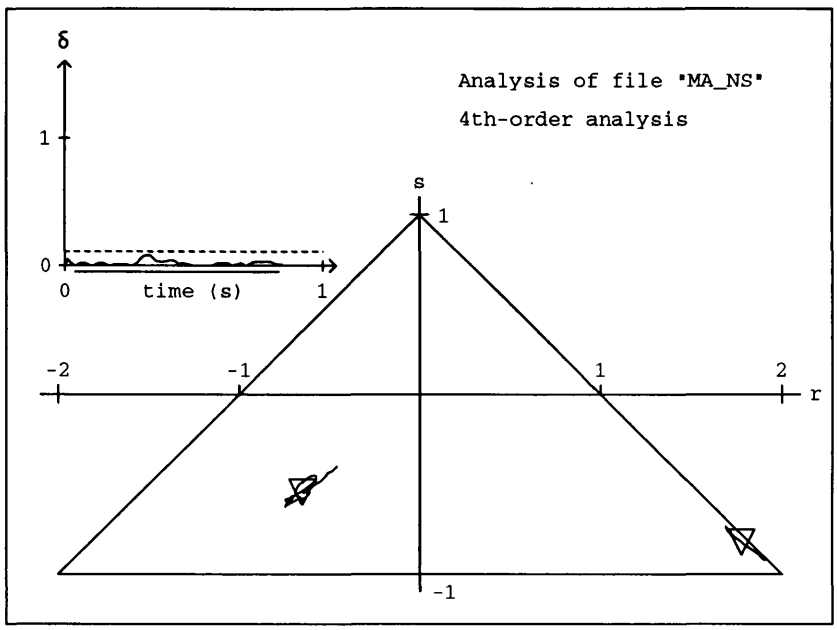

Figure 64: Analysis of /ma/ (speaker JT with added noise) using a 4th-order model.

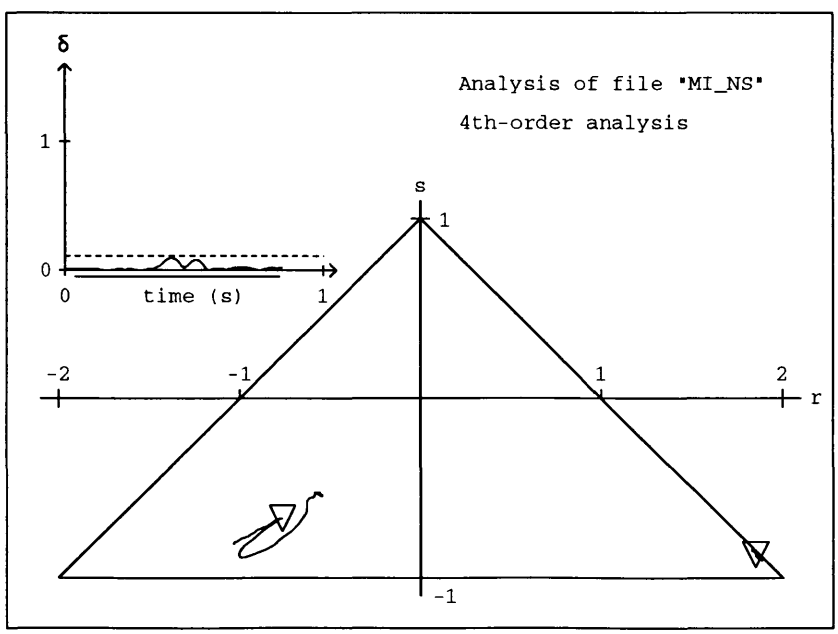

Figure 65: Analysis of /mi:/ (speaker JT with added noise) using a 4th-order model.

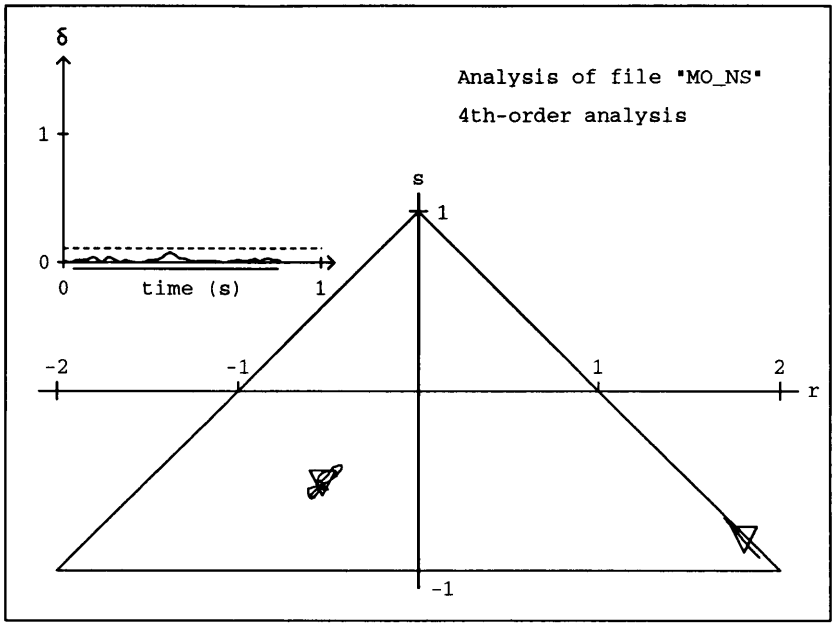

Figure 66: Analysis of /mo/ (speaker JT with added noise) using a 4th-order model.

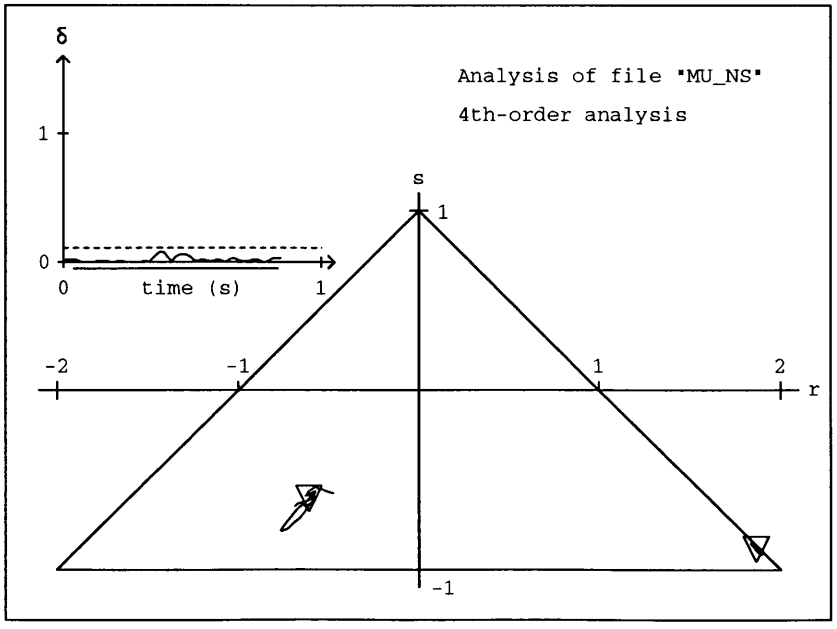

Figure 67: Analysis of /mu/ (speaker JT with added noise) using a 4th-order model.

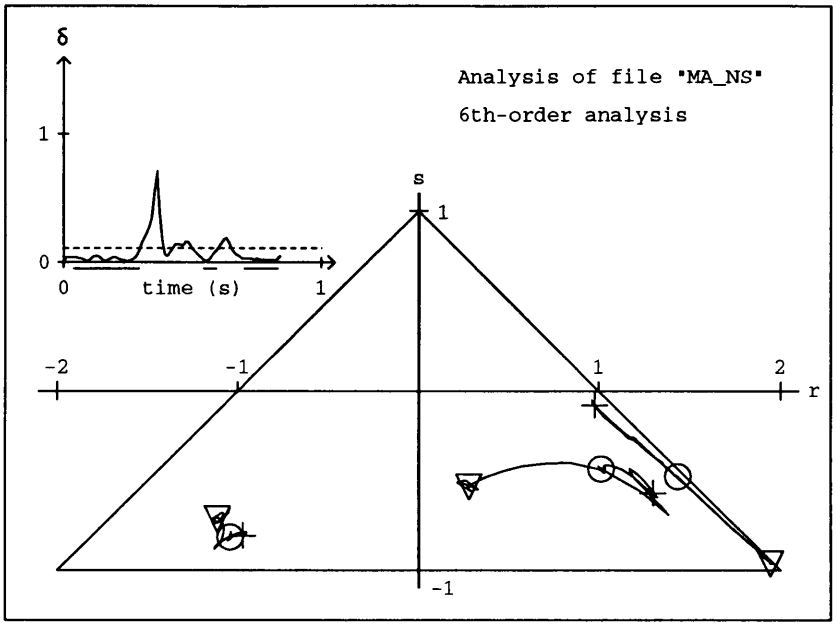

Figure 68: Analysis of /ma/ (speaker JT with added noise) using a 6th-order model.

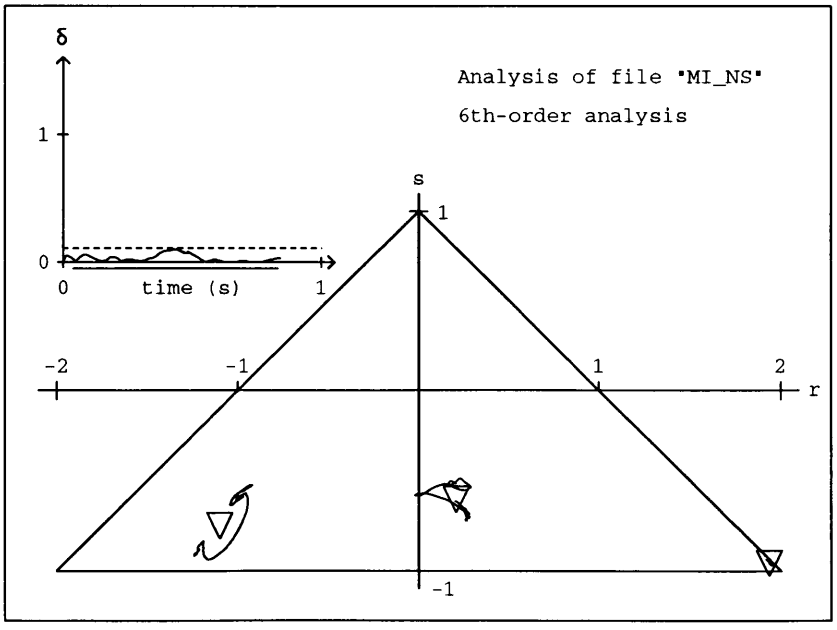

Figure 69: Analysis of /mi:/ (speaker JT with added noise) using a 6th-order model.

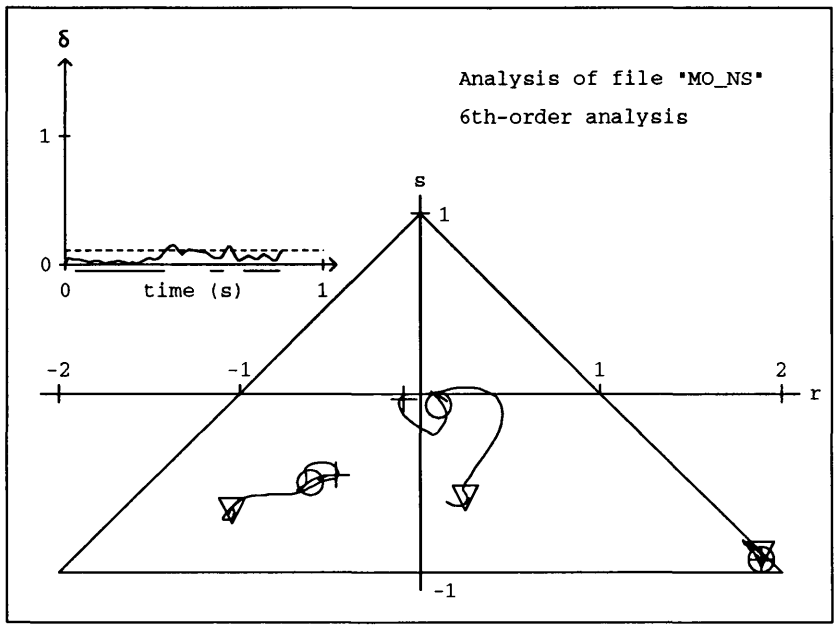

Figure 70: Analysis of /mo/ (speaker JT with added noise) using a 6th-order model.

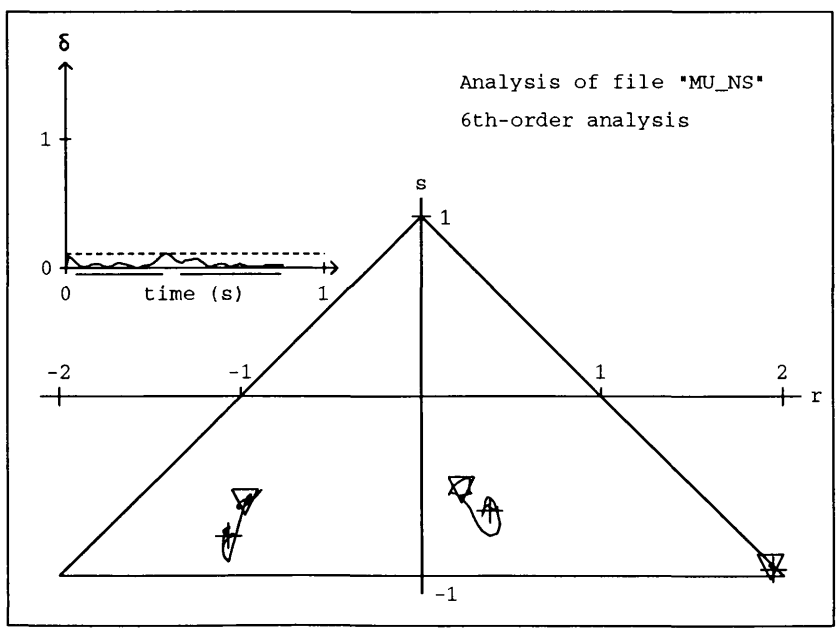

Figure 71: Analysis of /mu/ (speaker JT with added noise) using a 6th-order model.

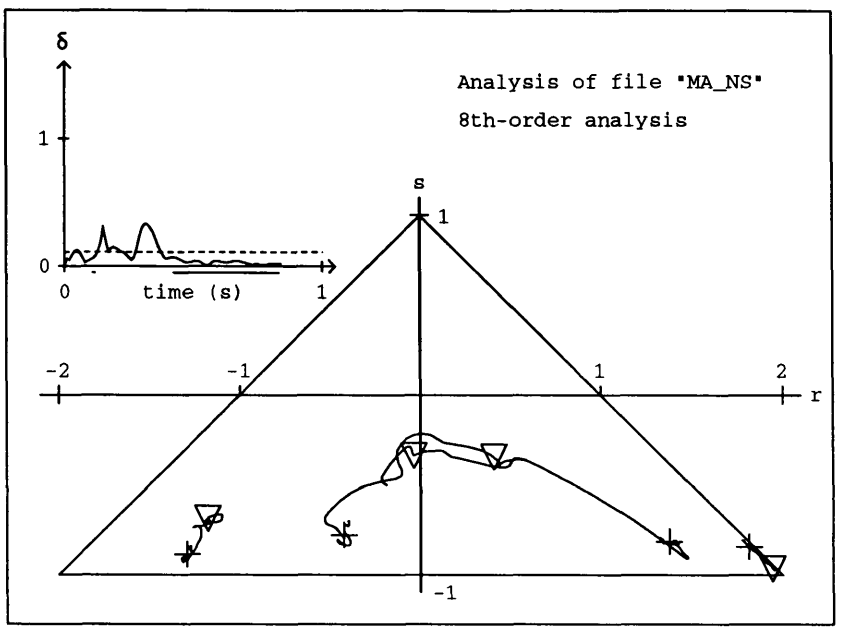

Figure 72: Analysis of /ma/ (speaker JT with added noise) using a 8th-order model.

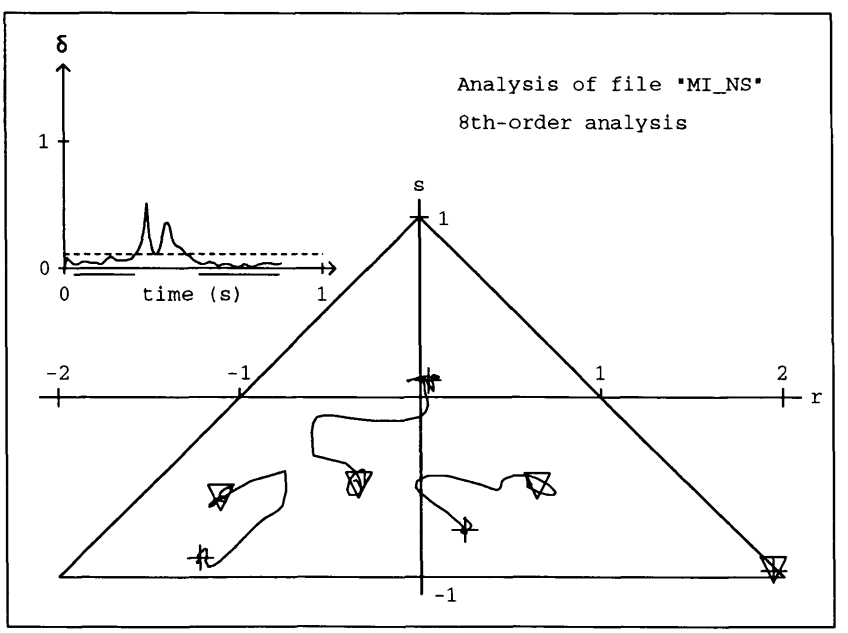

Figure 73: Analysis of /mi:/ (speaker JT with added noise) using a 8th-order model.

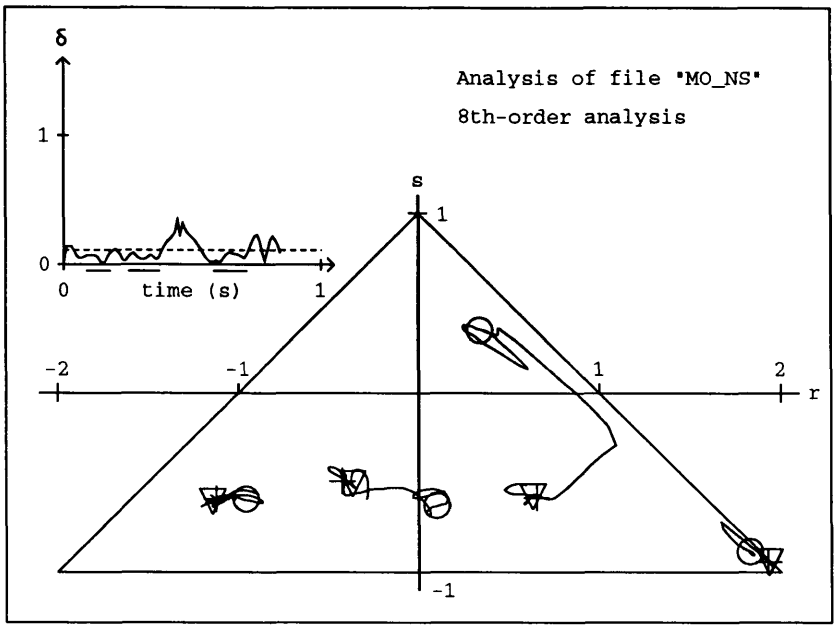

Figure 74: Analysis of /mo/ (speaker JT with added noise) using a 8th-order model.

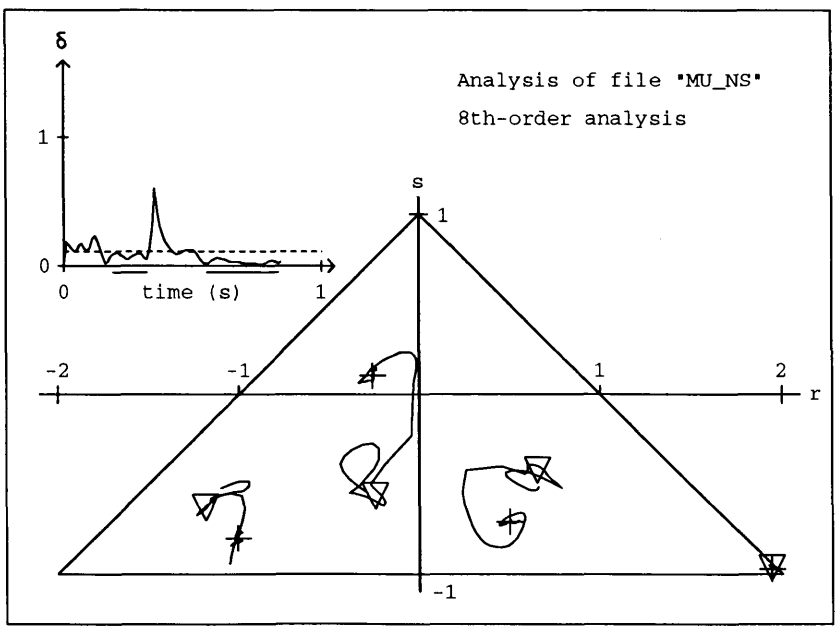

Figure 75: Analysis of /mu/ (speaker JT with added noise) using a 8th-order model.

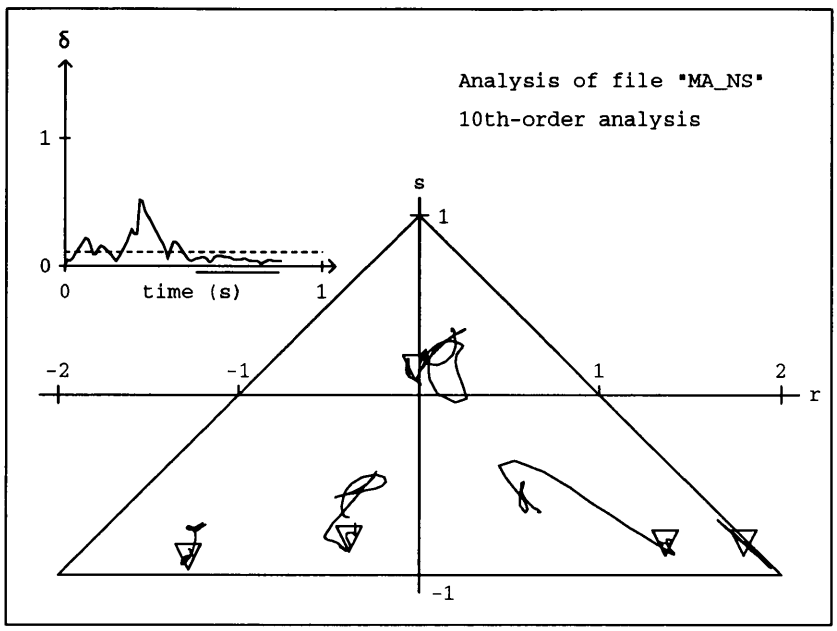

Figure 76: Analysis of /ma/ (speaker JT with added noise) using a 10th-order model.

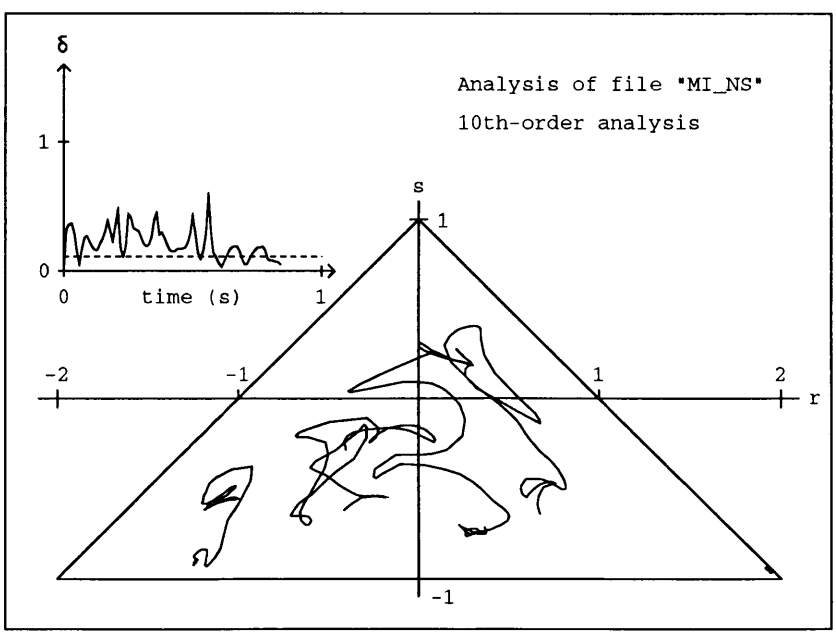

Figure 77: Analysis of /mi:/ (speaker JT with added noise) using a 10th-order model.

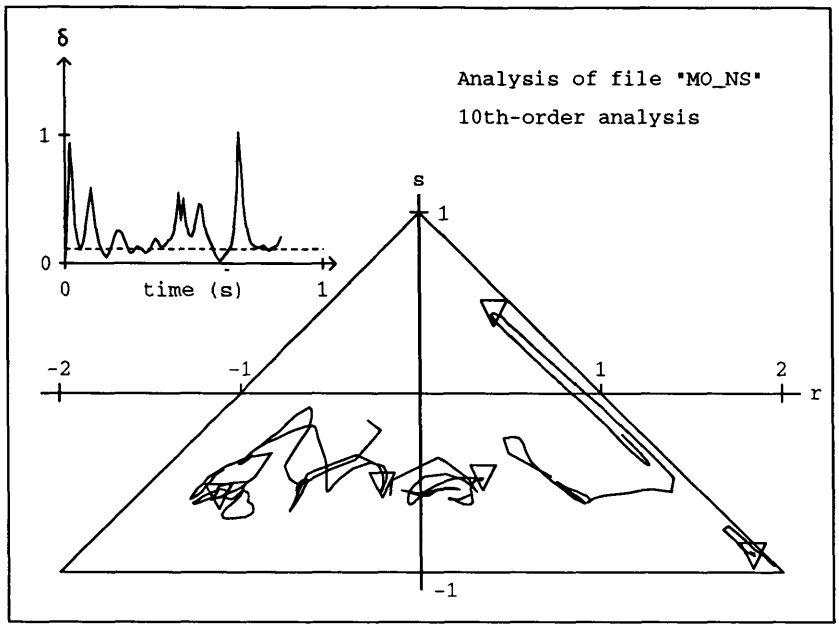

Figure 78: Analysis of /mo/ (speaker JT with added noise) using a 10th-order model.

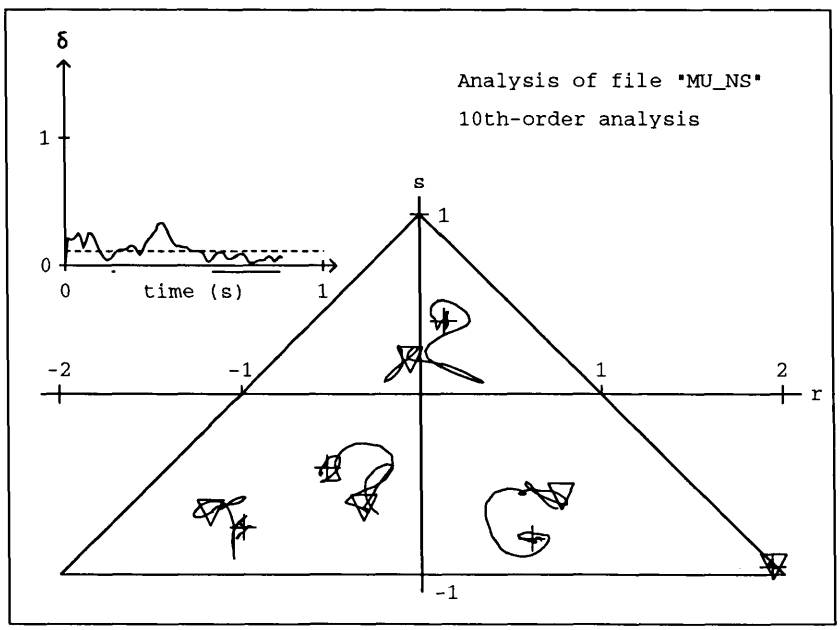

Figure 79: Analysis of /mu/ (speaker JT with added noise) using a 10th-order model.